PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE ARTES ESCÉNICAS

Ciclos sonoros: Una aproximación a la composición algorítmica e interpretación artística en SuperCollider

Tesis para obtener el título profesional de Licenciado en Música que presenta:

Leandro Marcelo Mellado Zarate

Asesora:

Zoila Elena Vega Salvatierra

Lima, 2023

Informe de Similitud

Yo, *Zoila Elena Vega Salvatierra*, docente de la Facultad de Artes Escénicas de la Pontificia Universidad Católica del Perú, asesor de la tesis de investigación titulada "*Ciclos sonoros: Una aproximación a la composición algorítmica e interpretación artística en el entorno de programación SuperCollider*", del autor *Leandro Marcelo Mellado Zarate* dejo constancia de lo siguiente:

- El mencionado documento tiene un índice de puntuación de similitud de 4%. Así lo consigna el reporte de similitud emitido por el software *Turnitin* el *01-nov.-2021*.
- He revisado con detalle dicho reporte y la tesis, y no se advierte indicios de plagio.
- Las citas a otros autores y sus respectivas referencias cumplen con las pautas académicas.

Lugar y fecha: Lima, *13 de junio de 2023*

Apellidos y nombres del asesor: *Vega Salvatierra, Zoila Elena*	
DNI: *29738283*	Firma 
ORCID: *https://orcid.org/0000-0002-6748-7648*	

Resumen

La presente investigación documenta los procesos de construcción de tres instrumentos digitales creados en el entorno de programación especializado en sonido llamado SuperCollider. Este software opera con principios de síntesis digital que genera sonido a través de la creación de algoritmos matemáticos. Los instrumentos propuestos corresponden a generadores de formas musicales que el autor denomina como ciclos sonoros. Estos consisten en eventos sonoros cíclicos que se repiten indefinidamente en el tiempo. Sus elementos sonoros se agregan de manera aleatoria dentro de un marco de parámetros musicales previamente designados como la altura, duración, ritmo y amplitud. Sin embargo, su construcción se genera en un proceso de programación digital, con un enfoque algorítmico a través de un código escrito que se encuentra documentado en la sección de anexos. Como resultado de un análisis de la construcción interna de los instrumentos digitales, del su funcionamiento y sus posibilidades de variación se evidencian las peculiaridades que presenta la utilización de este instrumento digital para la generación de sonido en tiempo real. A su vez, estos estudios proponen la viabilidad de utilización del instrumento dentro de la performance artística en vivo. Finalmente, el estudio presenta ventajas únicas frente a otras herramientas de creación sonora basadas en la tecnología analógica y digital del sonido, tales como sintetizadores modulares, secuenciadores, drum machines, DAWs.

Palabras clave: música algorítmica, composición algorítmica, síntesis sonora, SuperCollider, instrumentos digitales, ciclos sonoros.

Índice

Resumen ii
Índice iii
Índice de figuras iv
Introducción 1
Propuesta de investigación 5
Planteamiento del problema 5
Objetivo 6
Justificación 6
Estado del arte 8
Marco teórico 11
Capítulo 1: Síntesis digital 19
1.1 SuperCollider: funcionamiento básico del software 23
1.2. Amplitud Modulada: principio algorítmico de los ciclos sonoros 30
Capítulo 2: Construcción de instrumentos digitales en SuperCollider 38
2.1. Caso 1: Frecuencias ordenadas simétricamente 40
2.2. Caso 2: Multichannel con modificación del ritmo interno al cambiar las fases de cada elemento 43
2.3 Caso 3: Empleo de iteración en un instrumento 46
2.4 Resultados 51
2.4.1 Instrumento de estudio 1 (~est1) 52
2.4.2 Instrumento de estudio 2 (~est2) 54
2.4.3 Instrumento de estudio 3 (~est3) 56
Conclusiones 59
Referencias bibliográficas 63
Anexos 66

Índice de figuras

Figura 1. Flujo de información 20
Figura 2. Gráfica de ondas de sonido 21
Figura 3. Visualización de ciclos en onda de sonido 21
Figura 4. Ondas con amplitud 1 y 0.5 22
Figura 5. Formas de onda 23
Figura 6. Instrucción de reproducción de Onda sinusoidal escrita en SC 26
Figura 7. Asignación de variable a función 27
Figura 8. Envolvente ADSR 28
Figura 9. Incorporación de un envolvente ADSR 29
Figura 10. Amplitud modulada 31
Figura 11. Onda cuadrada de 2 Hz 32
Figura 12. Onda cuadrada con un duty cycle de 0.1 33
Figura 13. Onda cuadrada con distintas fases **¡Error! Marcador no definido.**4
Figura 14. Ondas moduladoras con fases diferentes 35
Figura 15. Algoritmo de iteración 36
Figura 16. Codigo del estudio 1 40
Figura 17. Onda triangular 42
Figura 18. Onda triangular con duty cycle 0.4 y elevada al cubo 42
Figura 19. Código de estudio 2 44
Figura 20. Onda sinusoidal, sus valores son menores a 0.4 la mayor parte del ciclo 45
Figura 21. Onda sinusoidal operada con una operación Booleana 45
Figura 22. Código de estudio 3 47
Figura 23. Onda sinusoidal elevada a 100 48
Figura 24. Reproducción del instrumento ~est3 49
Figura 25. Código de modificación de valores 50
Figura 26. Modificación de varios argumentos en una sola ejecución 51
Figura 27. Duración de las señales con un duty cycle de 0.2 53
Figura 28. Efecto producido por dos ondas con frecuencias cercanas 54
Figura 29. lthan 0.9: la onda cuadrada es más ancha en 0 y angosta en 1 55
Figura 30. Creación de polirritmias con ciclos diferentes 58

Introducción

Mi trabajo de investigación comprende un aspecto muy específico de mi producción como compositor. Considero la investigación artística como el medio más acertado para poder expresar textualmente las reflexiones conceptuales y teóricas que he generado acerca de la música como disciplina intelectual. Para esto, me propuse la tarea de escudriñar en las formas musicales de composición que utilizo en mi quehacer artístico para encontrar la manera de traducir mi proceso creativo más instintivo a un lenguaje textual y además académico. Con esta búsqueda he podido observar más detenidamente las corrientes reflexivas por las que mi propio entendimiento de la música ha navegado; observar en ellas procesos más específicos y tratar de entender por qué me funcionan. En la investigación artística, el creador no puede postular una forma de entender el arte como una verdad objetiva, y tampoco es mi intención pretender proclamar una. Me parece más interesante, más bien, poder analizar y describir diversas variaciones acerca de un proceso creativo concreto; documentar sus características, exponer sus peculiaridades y compartir ese conocimiento con la comunidad compositora y musical de medios tradicionales y nuevos. Espero que este trabajo plasme claramente mi propia forma de observar al sonido como una fuente reveladora de nuestra propia consciencia.

¿Cómo usar un looper?

El proceso creativo que pretendo abordar fue obtenido inicialmente desde la exploración y utilización de pedales de *looper* y algún instrumento musical como un lienzo en blanco. A este lienzo, se le agregan sonidos como pinceladas que forman una música particular, un espacio cíclico compuesto por cada uno de los elementos sonoros utilizados. Considero que este resultado no se logra utilizando un *looper* de la forma convencional porque una intención primaria es deslindarnos del tempo y no pensar en construir ritmos.

Pongamos como ejemplo que utilizo una pedalera *looper* y configuro la duración del ciclo de grabación en 10 segundos. A continuación, ajusto el nivel de retroalimentación (*feedback o decay*) en 100%, esto impide que las repeticiones generadas en el *looper* reduzcan su volumen progresivamente y, en cambio, mantengan su volumen al máximo por largos períodos. Con esto ya tenemos nuestro lienzo en blanco. A continuación, utilizo una guitarra para grabar sobre el *looper*. La manera de grabar sobre él es agregar solamente una corta nota por cada ciclo que registra el *loop*. En cada repetición del ciclo se introduce una nota distinta que se encuentre dentro de la escala mayor. Para su construcción no es necesario considerar una organización métrica que delimite un pulso fijo, ni intentar construir patrones rítmicos con la adición de notas. Más bien, cada nota se agrega de manera espontánea, tratando de llenar los espacios de silencio. La idea es repetir este proceso hasta eliminar total o parcialmente los espacios de silencio en la repetición. De esta manera, el lienzo resulta en un espacio lleno de sonidos aparentemente desorganizados.

Sin embargo, se ha creado lo que será llamado en adelante como ciclo sonoro. Si lo escucho lo suficiente, puedo encontrarle un sentido musical. En un proceso parcialmente indeterminado puedo encontrar un sentido que se genera por diversos motivos que se verán durante la investigación, pero creo que el principal está declarado por la naturaleza del *looper*: la repetición. Es en la repetición que nosotros encontramos sentido a los sonidos. Este ciclo sonoro me invita a descubrir un sentido en él, y a reconocer un discurso musical coherente. Puedo incluso identificar melodías interesantes, con características únicas, debido a que su construcción nunca tuvo una intención rítmica ni dirección preconcebida.

En procesos más convencionales de construcción melódica, existen dos elementos importantes: dirección y ritmo. Pero en el ejemplo que nos ocupa, ninguno de los dos fue planificado porque la misma forma de construcción nos exigía a no pensar la tarea de manera lineal, sino improvisada, espontánea.

En lo que respecta al ritmo, aunque inicialmente no existe uno predefinido, finalmente, ocurre de manera aleatoria, y se puede lograr percibir una secuencia de notas con movimiento rítmico. En lo que respecta a la altura, en nuestro ejemplo, las notas que tocamos con la guitarra fueron solamente notas de una escala mayor, de esta manera establecemos un repertorio determinado de alturas, por lo que no será difícil sentir el ciclo sonoro como tonal. Sin embargo, la dirección melódica se forma aleatoriamente. Aunque se encuentre limitado al rango de una escala mayor, considero que este acotamiento de las notas a una escala cualquiera es suficiente para detectar un espacio sonoro con un discurso identificable, pero al mismo tiempo no convencional.

Ejemplos de ciclos sonoros:

1. Guitarra en un ciclo de 10 segundos
2. Guitarra en un ciclo de 4 segundos
3. Síntesis en SuperCollider 1
4. Síntesis en SuperCollider 2

El pequeño ejemplo muestra en términos generales el resultado de la creación de un ciclo sonoro. se pueden observar ciertas similitudes entre los ejemplos: todos construyen un ritmo y armonía reconocibles y únicos, a los cuales hemos llegado aplicando un principio de aleatoriedad. Las melodías corresponden a una serie de notas que surgieron aleatoriamente, pero esas notas corresponden a cierta escala. Tampoco existió una pre-composición del ritmo; sin embargo, siempre se termina formando alguno, por más que no sea técnicamente exacto o simétrico. Los dos primeros ejemplos representan una aproximación a través del uso de un *looper* y los dos siguientes a una implementación de los ciclos sonoros en SuperCollider. Durante el tiempo de composición de ciclos sonoros en el que usaba un pedal de *looper*, me

surgieron algunas preguntas con respecto a la capacidad de manipular los elementos (notas) que conforman un ciclo sonoro, porque me topé con una primera limitante, el ciclo sonoro se ha grabado como audio a través de un DAW usando un efecto de *loop*. Este resultado se puede utilizar de muchas maneras dentro de una composición. Sin embargo, si se piensa en el uso de esta forma como parte de una interpretación en vivo, la forma de creación del ciclo se vuelve repetitiva y larga si es que se construye en tiempo real, además no se pueden aplicar variaciones a los elementos individualmente. Aunque yo mismo he utilizado la forma de *añadir nota por nota* para crear un ciclo sonoro como una forma musical estructural, recurrir solo a esta, se tornaría repetitiva para mí, debido a su construcción lenta, que se podría tornar aburrida o muy reusada para un oyente.

Como ejecutante, y con el interés de interpretar estas formas musicales en vivo, he buscado otra forma de generación de ciclos sonoros para poder adaptar las formas cíclicas en formatos que me permitan crear un ciclo sonoro de manera instantánea, sin tener que recurrir a la grabación manual de capa por capa. modificar los elementos y con ello poder darle más forma y variación a mi propuesta. Recurrí entonces a su implementación un software de programación de sonido llamado SuperCollider, en donde, he podido construir instrumentos que generan ciclos sonoros. Además, puedo generar estos ciclos y manipularlos de manera orgánica y a tiempo real, lo que me permite interactuar directamente con el sonido en medio de una performance. Por ejemplo, puedo modular las notas de la escala mayor que utilicé en el ejemplo anterior hacia otra escala, o puedo cambiar los ordenamientos de los elementos que suceden, o incluso modificar los ritmos que se generaron siempre a base del uso de la aleatoriedad.

El desarrollo de la investigación consistirá de un primer capítulo en donde se detallará el diseño técnico de la generación de ciclos sonoros a través de un entorno de programación de sonido haciendo uso de las propiedades del sonido aplicadas a la síntesis y del lenguaje

técnico del software SuperCollider. En el segundo capítulo, se realizarán 3 estudios de casos que corresponden a la creación de un instrumento digital. Se documentarán los procesos de conceptualización y realización de las formas con el propósito de visibilizar la capacidad de manipulación que alcanzan en cada una. Y, finalmente, en las conclusiones se discutirán los resultados de estudio con el interés en exponer las capacidades musicales que tienen dichos instrumentos digitales.

Propuesta de investigación

La presente tesis propone la documentación del proceso de construcción un instrumento que genere ciclos sonoros en el software SuperCollider. Se trata de visibilizar las capacidades interpretativas que el programa permite para aplicar su uso en una interpretación en vivo. De esta manera, se establecerá una relación teórica entre las propiedades sonoras, computaciones y compositivas. Asimismo, se compartirá una visión particular sobre la composición musical basada en el uso de herramientas digitales, la cual implica una perspectiva y utilización de los objetos de SuperCollider peculiar para la construcción de un instrumento digital que genere ciclos sonoros. Dicha aproximación se plantea para compartir el conocimiento de uso de softwares aplicados a la composición musical, que podrá ser aplicada de diversas maneras o en otros entornos artísticos por cualquier interesado.

Planteamiento del problema

Los ciclos sonoros son una forma de creación sonora que consiste en la añadidura de elementos cíclicos agregados casi aleatoriamente. La manera inicial de creación de esta forma musical consistió en un proceso de grabación de instrumentos o sonidos con un pedal o efecto de *looper.* Sin embargo, esa aproximación presenta una barrera que limita el desarrollo de los ciclos sonoros como una forma de creación en tiempo real, debido a la naturaleza de un pedal

de *looper* o de un DAW que funciona con grabaciones que se realizan en el momento. El presente estudio propone la creación de ciclos sonoros mediante un software de síntesis digital que genera sonidos por medio de la programación. Esta construcción nos permite controlar los de elementos sonoros individualmente, con el fin de poder manipular y modular los elementos del ciclo sonoro en un entorno interpretativo a tiempo real.

Objetivo

El objetivo principal de esta investigación es identificar, exponer y categorizar, de manera documentada, las posibilidades de manipulación y las aproximaciones novedosas de los instrumentos digitales que desarrollé con el fin de sostener la utilidad de los ciclos sonoros en un entorno de interpretación en vivo. Como objetivo secundario se encuentra la exposición de la construcción de los instrumentos en donde se describe la función de los objetos y métodos usados para su desarrollo.

Justificación

Tomando en cuenta los objetivos de la investigación, considero necesario documentar las prácticas artísticas, procesos creativos, y los proyectos realizados de los artistas contemporáneos en el Perú por las siguientes razones. En primer lugar, la documentación académica deja impresa una práctica humana valiosa. En el caso del desarrollo de la música en el Perú, existe un vacío de literatura académica. Esto ha ocasionado un desconocimiento parcial de las prácticas culturales musicales del pasado y del presente, y muchas veces ha dificultado el trabajo musicológico, pedagógico y creativo de hoy en día. Es relevante escribir acerca del trabajo artístico para que, de esta manera, quede constancia de su existencia y de sus procesos. En segundo lugar, este trabajo busca detallar una serie de procesos creativos aplicados a la composición y ejecución musical a través de un instrumento construido digitalmente. Una documentación procedimental detallada será de utilidad para la comunidad

compositora puesto que la presente investigación propone, dentro del campo de la composición algorítmica, nuevas formas de aproximación a la composición musical. Ya se ha visto una integración interdisciplinaria importante en proyectos artísticos que incorporan nuevas tecnologías en su realización. Finalmente, considerando el estado del desarrollo musical en el Perú, es importante escribir acerca de la adopción de nuevas tecnologías en el arte y en la creación de este. Creo que el arte siempre debe tener una mirada positiva frente a los recursos disponibles, y el desarrollo global en tecnologías digitales se ha presentado como una opción muy llamativa para las nuevas generaciones artísticas. En este sentido, una investigación que discuta estos (no tan) nuevos paradigmas, posiciona a la producción artística nacional, un poco más, en la vanguardia.

Estado del arte

Mucha de la investigación realizada acerca de temas similares se centra en la discusión de las nuevas tecnologías como parte de los procesos compositivos. Estas discusiones las observamos en trabajos como el de Jessica Rodriguez, que menciona ciertos sistemas algorítmicos que surgen como procesos compositivos. (Rodriguez, 2017, p.26). Se encuentra también el trabajo de Carlos Gutiérrez, que parte de reflexiones acerca de la utilización de nuevas herramientas en la práctica artística. (Gutiérrez, 2014, p.10). También me apoyaré del trabajo de la reflexión estética propuesta por Joan Bagés, que busca describir un valor estético en el uso de herramientas tecnológicas (Bagés, 2011, p.8). El trabajo de Xavier Berenguer tiene una intención de reflexión acerca del uso de las nuevas tecnologías en el arte (Berenguer, 2002). El trabajo de James McCartney del 2002 advierte una introducción a la programación musical con el software que se incluirá en la descripción procedimental de la tesis, SuperCollider (McCartney, 2002). Se encuentra progresivamente, un trabajo acerca de prácticas modernas de manipulación de software de programación (Collins, McLean, Rohrhuber & Ward, 2003, p.9). El uso de estas herramientas se profundiza con el escrito de Wesler, quien reflexiona sobre la investigación artística, desde una perspectiva personal que se trata su proceso creativo musical utilizando herramientas electrónicas (Wesler, 2018). Finalmente, el trabajo de Jaime Lobato (2021), quien propone formas nuevas de creación sonora a través del sonido generado por computadoras químicas.

SuperCollider es un entorno de programación aplicado al diseño de sonido a través de instrucciones matemáticas llamadas algoritmos. Es un software que permite crear sonidos sin una fuente previa. Su uso está repartido en varias áreas como la educación, la síntesis digital, la composición algorítmica, la construcción de instalaciones, la performance, entre otros. Dentro del área de la performance, el *live coding* se ha convertido un modelo bastante difundido

en la cultura electrónica, como dice Thor Magnusson (2014) en su artículo que hace un recuento conceptual del *live coding* a diez años de sus inicios. Esta práctica consiste en la performance de una o un grupo de personas que generan música a través de la construcción de códigos en tiempo real utilizando mayormente laptops. Es usual que en estas prácticas se proyecte la pantalla de la computadora en donde el *performer* escribe códigos y los modifica.

Con respecto a trabajos especializados en el software SuperCollider, encontramos aquellos que documentan aproximaciones de técnicas de síntesis de sonido, los cuales exponen novedades de aplicación. Como el trabajo de Paul Rhys (2016), quien desarrolla un software diseñado para generar síntesis granular en SuperCollider con un rico potencial compositivo. También tenemos el trabajo conjunto de Anna Xambó, Alexander Lerch y Jason Freeman (2019) quienes proponen un método de recopilación de datos a través del análisis de audios para la aplicación de dichos datos en un entorno de *live coding*. Asimismo, se encuentran escritos con enfoques educativos sobre la síntesis de sonidos. Como por ejemplo el blog de Jon Woo Park, quien comparte perspectivas compositivas electrónicas propias exploradas en SuperCollider. De manera parecida, Oscar Recarte, investigador sonoro peruano, comparte en su blog técnicas de síntesis digital de sonido implementadas en SuperCollider. Otras fuentes de trabajos tienen que ver con la enseñanza de uso del software para estudiantes, como el repositorio de videos de Eli Fieldsteel (2012), director de los estudios de música experimental en la universidad de Illinois.

Por otro lado, es importante mencionar trabajos que discuten procesos compositivos musicales similares y que sirvieron de cimiento e influencia para el desarrollo de la forma musical presentada en esta tesis. El trabajo de Barney Stevenson es muy importante, pues documenta técnicas y realiza análisis musicales acerca del trabajo de Brian Eno (Stevenson, 1997). El trabajo de Iannis Xenakis, quien realiza una autoetnografía acerca de sus procesos creativos en varios de sus trabajos experimentales (Xenakis, 1978). Steve Reich (1968)

también realiza una documentación acerca de su proceso creativo con respecto a trabajos compositivos propios. Finalmente, en el texto de John Cage, *El futuro de la música* (1937), él comparte reflexiones acerca de la práctica musical del futuro, nuevas prácticas que ahora se encuentran disponibles para todo el mundo a través de una computadora.

Junto a estas fuentes me es más viable encajar mi investigación, que discute procesos de creación artística con la utilización de software de programación de sonido, en un marco autoetnográfico reflexivo acerca de propuestas artísticas propias. Los párrafos albergan dos grupos de fuentes, el primer grupo consiste en fuentes cuya discusión abarca investigaciones, reflexiones y propuestas que acogen la tecnología de las computadoras como una herramienta tecnológica que puede ser usada para la creación artística y que mantengan un valor estético único y válido. En el segundo párrafo, el grupo de fuentes abarca la documentación sobre perspectivas compositivas personales, algunas realizadas en el estilo de redacción de autoetnografía. Es en estos trabajos en donde se observa valor en la (auto)documentación que discuta procesos creativos individuales, pues en estas documentaciones se puede encontrar material inspirador para la creación (Bartleet & Ellis, 2009, p. 11)

Marco teórico

Brian Eno postula que su proceso creativo enfocado a la música ambiental inicia paralelamente que la creación de la pieza musical. Introduce el término composición *dentro-del-estudio*, donde uno ya no tiene que ir al estudio musical con una idea preconcebida. Sino, con una mínima idea de estructura, o con ninguna idea en particular, y es ahí, en el estudio, donde uno va explorando las herramientas propias del estudio con las que se pueden probar distintos parámetros, y una vez así, uno inicia el trabajo de composición (Eno, 1979).

Wesler es quien nos aproxima a la experiencia del proceso creativo aplicada a la composición sonora con la utilización de tecnología electrónica. Propone en primer lugar, que la máquina es una herramienta con la que activamente el creador se irá relacionando durante su proceso creativo. Postula que la máquina genera una ruptura, generación o modificación de ideas del creador, debido a que, durante la exploración, el creador está constantemente evaluando las posibilidades que la máquina le ofrece, por lo que puede encontrar una idea o probar un resultado que le guste debido a varios factores. Muchas veces, esto puede cambiar el rumbo de una composición. También menciona que la interacción con la máquina sigue una comunicación lúdica de acuerdo al proceso de aprendizaje y dominio de la máquina, este sirve como una fuente de inspiración sujeta a las posibilidades de manipulación. Almighty Tabuena (2018) recalca un punto importante sobre la aplicación de la aleatoriedad en la composición de música aleatoria, que implica el uso de elemento sonoros que pueden variar y no ser necesariamente los mismos durante la performance. Estas técnicas iniciadas en el siglo XX por pioneros como John Cage, hoy en día se adoptan en el uso de la computadora para generar esos valores aleatorios, automatizando el proceso de la generación de patrones aleatorios, con lo que es más fácil la utilización de este recurso. Otro punto a destacar es la economía de recursos. Es importante delimitar las herramientas virtuales que se utilizarán y el

proceso computacional que implica su uso, ya que dependemos también de los límites de nuestros aparatos para trabajar con un flujo de trabajo constante. Finalmente, Wesler hace mención de los tres responsables del resultado de un proceso creativo que son: el software, el fabricante y el creador (músico, compositor, etc). El software y el fabricante delimitan las posibilidades máximas a las que el creador puede llegar para la ejecución de su proceso creativo.

El proceso creativo que propongo documentar para la presente se conformaría principalmente por dos conceptos de concepción musical. El primero hace referencia al concepto *Ambient music* introducido por el compositor Brian Eno, el segundo se basa en la práctica del *computer music* como género musical, que utiliza la computadora como instrumento principal para la composición de música.

El término ambiental, dentro del imaginario musical, tiene interpretaciones que toman en consideración el espacio físico. Para John Cage, el significado es tan sencillo como los sonidos que rodean el lugar en donde uno se encuentra. La Real Academia de la Lengua Española lo distingue como un adjetivo que rodea algo o a alguien como elemento de su entorno. Para los ingenieros de sonido el ambiente hace referencia a la dimensión espacial atribuida a un sonido con algún tipo de efecto de *delay* o reverberación (Tamm, 1988). Para Brian Eno, la música ambiental es aquella que puede teñir la atmósfera en donde se reproduce, música que rodea al oyente con un sentido de espacialidad y profundidad que abarque todos los espacios posibles (Tamm, 1988). Eno menciona que la música ambiental debe ser capaz de acomodarse en diferentes niveles de escucha activa sin forzar una en particular, debe ser tan olvidable como interesante (Lysaker, 2017).

Brian Eno publica *Ambient 1: Music for Airports*. Este trabajo se autodenominó explícitamente como un álbum de música ambiental, y representa una continuación de su exploración de música generativa. El concepto de composición de las piezas nos sirve como

referencia directa en la construcción de los ciclos sonoros. Como vemos en el artículo en la página de Reverb Machine (2019), La aproximación compositiva de Eno consiste en la reproducción de varias grabaciones realizadas en cinta magnética. Las grabaciones representan interpretaciones de frases de una pequeña cantidad de notas en donde todas tienen diferentes tiempos de duración. Todas las cintas se reproducen al mismo tiempo una y otra vez. Debido a la duración de las grabaciones la relación entre cada grabación se va desplazando, cada vez, las frases se intersecan de manera diferente y en el resultado pueden surgir frases o motivos nuevos debido a la relación de las grabaciones. Eno menciona que para la primera pieza "*1/1"* utilizó 22 grabaciones, donde 8 de ellas son grabaciones de un coro cantando una sola nota por 10 segundos, y 14 corresponden grabaciones de un piano, a veces solo son una o dos notas (Deconstructing Brian Eno's "Music for Airports", 2019).

La segunda perspectiva se localiza en un centro cultural en desarrollo, aquel que involucra la tecnología digital como fuente principal en la ejecución, composición, e investigación sonora, pues, las herramientas y técnicas consisten en la utilización de software dedicado a la creación, grabación y manipulación de sonidos. Debido a la amplitud de este tema, es necesario guiar al lector por los caminos específicos en donde yo he desarrollado el proceso creativo. Asimismo, será pertinente delinear la evolución histórica de los procesos que influyeron en la existencia de los recursos digitales actuales que dispuse para mi investigación personal.

La tecnología aplicada a las artes puede generar cuestionamientos y discusiones en torno al valor artístico y estético que podría ponerse en juego cuando la herramienta principal de composición consiste en la utilización de una máquina que facilita procesos para el creador. Sin embargo, un adelanto tecnológico, como en este caso, una computadora, no condiciona necesariamente el proceso creativo. Este representa un instrumento útil para el creador, facilita distintos procesos mecánicos, y aún más importante, hace posible generar

otros procesos, que no podrían darse si no existieran dichos avances tecnológicos. Entonces un instrumento tecnológico soluciona problemas para el creador: por un lado, realiza procesos que podrían ocuparle mucho tiempo, limitando el momento del proceso creativo que dispone para la creación. Por otro lado, hace posible la exploración dentro de un marco de posibilidades que solamente existen por la tecnología utilizada. El cómo se utiliza la tecnología ya corresponde en su totalidad al creador, y es en este proceso en donde el valor de la creación se definirá estéticamente en mayor medida.

La tecnología se erige como un motor de creatividad. Joan Bages menciona que las nuevas tecnologías musicales nos revelan la confluencia entre factores científicos, tecnológicos, sociales, filosóficos y musicales de nuestra época. Él escribe una reflexión sobre una tendencia latente durante el siglo veinte que involucra la individualización en las expresiones humanas, entre ellas, la música. Así como en filosofía o psicología se empieza a hablar de la atomización del individuo, y en la física aparece la física cuántica, en la música se establece una tendencia hacia la búsqueda de la unidad mínima del sonido, que busca llegar a su interior (Bages, 2011, pp. 3-4). Menciona, por ejemplo, al serialismo como técnica que fragmenta las notas y los parámetros musicales. Además de esto, la exploración de las técnicas de grabación y la síntesis sonora surgen como potenciales propuestas de individualización del sonido, pues ellas se enfocan en la obtención de timbres y texturas específicas de acuerdo a diferentes procesos.

La aparición de la grabación significa un inmenso avance para la humanidad, en diversos ámbitos, y esta fue evolucionando desde sus inicios con el fonoautógrafo y el fonógrafo. Sin embargo, la grabación en cinta magnética implica un hito dentro de la composición musical, pues fue el acontecimiento que da nacimiento a la música concreta, la cual explora el ensamble y la manipulación de sonidos grabados en una cinta que se podía recortar e insertar en otro lado. Este hecho separa al intérprete del instrumento musical, lleva

al compositor a trabajar directamente en un estudio, a modelar el resultado final de su composición en una grabación sin tener que considerar la interpretación de su obra a manos de un músico. La figura del compositor se asemeja a la de un artista visual (Keislar, 2009, pp. 16-17).

La música electrónica otorgó mayor capacidad de manipulación sobre los parámetros del sonido. Mientras la música concreta se enfocó en la manipulación sobre sonidos grabados del mundo real. La música electrónica se enfocó en generar sonidos desde una construcción que partía del uso de la acústica, como la utilización de ondas sinusoidales o ruido. Los parámetros de altura, duración y amplitud se podían controlar arbitrariamente y combinar de maneras imposibles para instrumentos musicales tradicionales (Keislar, 2009, p. 18).

Con esto, me gustaría empezar a hacer referencia a la evolución y desarrollo de las tecnologías musicales, desde un plano contextual, haciendo mención a los elementos que significaron un avance tecnológico que permitió más capacidad de manipulación a los creadores sonoros. Mark Vail menciona que el primer sintetizador real se termina de construir en 1929 y es diseñado por los franceses Edouard Coupleux y Joseph Givelet, Este sintetizador se llamó “Automatically Operating Musical Instrument of the Electric Oscillation Type” y funcionaba con tubos de vacío que podían controlar la altura de los sonidos automáticamente. La automatización se interpretaba con una cinta perforada, los huecos hechos en la cinta correspondían a las automatizaciones de los parámetros del sintetizador (Vail, 2014 p. 168).

La siguiente invención tecnológica importante corresponde al instrumento electrónico Mark I de la RCA, desarrollado por los ingenieros electrónicos Harry F. Olson y Herbert Belar. Se desarrolló desde la década de los 40 hasta 1955 y corresponde al primer instrumento denominado expresamente sintetizador. El Mark I representaba un instrumento complejo pues usaba 12 osciladores para producir semitonos y que generaba ondas con forma

de dientes de sierra con múltiples armónicos, también admite el uso de filtros para poder realizar síntesis sustractiva. El diseño de los timbres y características del sonido se realizó también con el uso de cintas de papel perforadas que podía contener hasta 36 columnas de código binario, lo que permitió una capacidad de manipulación muy completa y variada. Sin embargo, la composición de música a través del Mark I tomaba varias sesiones de automatización de cada canal de osciladores por lo que su ejecución no podía ser a tiempo real.

Entre 1963 y 1970 se produce el primer sintetizador de Don Buchla, el Buchla 100 series consiste en un sintetizador modular que podía contener hasta 25 módulos de efectos. Otra característica importante es que posee un teclado sensible a la presión del tacto cuya respuesta se podía asignar a varios parámetros del sintetizador, como filtros o modulación de altura. También, se considera de los primeros secuenciadores de sonido, permitiendo al usuario diseñar una secuencia de notas que van generando un ciclo que se repite a una velocidad constante designada por un reloj interno.

El funcionamiento de estos sintetizadores era posible gracias a una tecnología analógica, en la cual, los procesos de síntesis se llevaban a cabo gracias a impulsos de voltaje que modulaban la afinación de las notas y los parámetros de los sintetizadores. Es a finales de los años 70 que esta tecnología se ve reemplazada por la tecnología digital. El Synclavier I, creado por la New England Digital Corporation, fue el primer sintetizador digital que se dispuso a la venta comercial. Consiste en un dispositivo generador de síntesis aditiva y también realizaba frecuencia modulada, un proceso de síntesis que no había sido dispuesto para uso comercial anteriormente, lo que ofreció una gama de nuevos sonidos a los usuarios. Al ser digital, el Synclavier I funcionaba con un software que cargaba información desde un disquete de donde recogía la información de funcionamiento del sintetizador.

Las computadoras juegan un papel importante en el desarrollo de la electrónica aplicada a la música. En 1949, empieza a operar la CSIRAC, la primera computadora digital de Australia, que fue también la primera máquina en reproducir música por sí misma. Este se podría considerar el inicio de la *Computer Music*. El término *Computer Music* se empieza a propagar durante la segunda mitad del siglo XX dentro del marco del arte occidental, la música estaba tomando diversos caminos y transformaciones debido a la exploración en el serialismo, la música aleatoria, el *noise*, y a la música electrónica. Y dentro de ella, podemos considerar dos aspectos en la *Computer Music*, según Douglas Keislar, podemos entender, por un lado, la *Computer Music* como un género musical, en donde la computadora forma parte de la composición, performance, o realización sonora. Por otro lado, se le entiende como una disciplina técnica, que explora y profundiza usos y avances tecnológicos para aplicarlos en relación a la creación sonora (Keislar, 2009, pp. 11-13).

Los primeros desarrollos tecnológicos que permitieron generar sonido aplicado a la composición en una computadora digital ocurrieron en los laboratorios Bell en la década de los 50. Fue el ingeniero electrónico Max Matheus quien inventó la síntesis de sonido digital, que significa la construcción de sonido a base de sonidos generados por un aparato digital. El primer software de síntesis de sonido, *MUSIC I*, se produjo en 1957 a cargo de Matheus y colegas. Es importante destacar que el proceso computacional era muy demandante para los dispositivos disponibles en la época, por lo que el proceso de síntesis no se podía oír a tiempo real, sino se debía esperar a exportar la información a una cinta magnética para posteriormente oír el resultado. Es desde la década de los 70 que la síntesis en tiempo real se vuelve una realidad, gracias a los avances tecnológicos de la época. De todas formas, la capacidad de procesamiento también es proporcional a la cantidad, complejidad de los procesos que se soliciten a la computadora o la cantidad de voces se están reproduciendo simultáneamente. Hoy en día es posible realizar la mayoría de los procesamientos sonoro en

tiempo real gracias a la capacidad tecnológica de los dispositivos actuales (Keislar, 2009, pp. 18-22).

Capítulo 1: Síntesis digital

Con este capítulo, además de exponer las maneras de documentar esta investigación, voy a introducir conceptos necesarios para que el lector se familiarice con la perspectiva de aplicación de este proceso creativo que introduce los ciclos sonoros. Estos primeros conceptos corresponden a un conocimiento general de la síntesis de sonido, aplicables en distintos entornos, tales como en el uso de sintetizadores modulares, funciones matemáticas, *plugins* y otros *softwares* de síntesis de sonido. En este caso, se trata de un proceso que involucra, además, el uso de un lenguaje de programación aplicado a la generación de sonido por procesos digitales de síntesis de sonido. Para su entendimiento será necesario abordar, de manera concisa, las propiedades del sonido, pues estas representan el pilar fundamental para realizar prácticas de síntesis. Seguidamente, se deben conocer algunos conceptos de operatividad corriente y sintaxis del software de programación de sonido SuperCollider, cómo estos operan las propiedades de sonido y cómo se pueden manipular, por medio de la programación, para lograr un resultado creativo, musical y alterable. Esta explicación no pretende introducir conceptos originales desarrollados por mí en la investigación; sino, es una mera introducción a los lectores no familiarizados con estos conceptos de entendimiento del sonido matemáticamente, necesarios para la utilización del software. Posteriormente, con un entendimiento previo de las herramientas a disposición de los usuarios, se expondrán las aproximaciones personales que llevé a cabo en el software en mención, las cuales constituyen la lógica principal de funcionamiento de los ciclos sonoros implementados digitalmente a través de la programación y su viabilidad de variación.

Cuando realizamos síntesis digital de sonido involucrando un software de programación, nosotros construimos un algoritmo matemático con una serie de instrucciones específicas que generan un torrente de información de código que es interpretado por el programa, y posteriormente esa información se dirige a la tarjeta de sonido de nuestro

ordenador, produciendo así un sonido audible (Cipriani & Giri, 2010), tal como vemos en la figura 1.

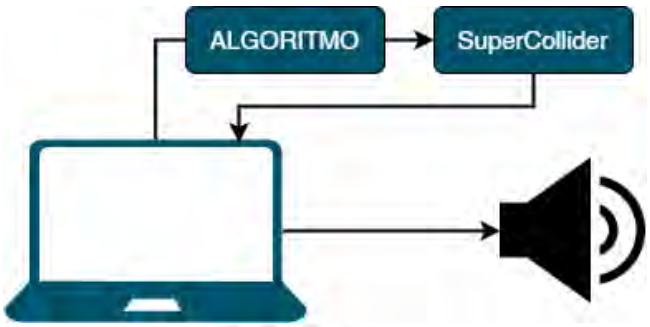

Figura 1. Flujo de información

En la práctica de síntesis se valora al sonido como un fenómeno físico que existe gracias a la vibración de la materia en un medio, como en el aire o en el agua. Las vibraciones de las partículas que oscilan en el aire a una gran velocidad son captadas por nuestros oídos y entendidas como sonido. Por medio de la práctica de síntesis se busca replicar este principio de oscilación para obtener sonidos generados de manera artificial, sin el uso de algún material acústico, sino amparado de la electrónica o, como en este caso, de la tecnología digital. La síntesis de sonido trabaja con las tres propiedades fundamentales del sonido para lograr modelar el sonido. La primera es la frecuencia, que determina la altura de un sonido, por ejemplo, podemos distinguir si un sonido es más agudo o más grave. Cuando hacemos referencia a la producción de un sonido en un medio como el aire, advertimos que las partículas del aire serán perturbadas y generarán oscilaciones en el aire. Usando como ejemplo una cuerda, cuando nosotros pulsamos la cuerda, ésta oscila hacia adelante y hacia atrás, y las partículas se mueven siguiendo ese cambio de dirección. Es de acuerdo a la velocidad de oscilación de las partículas que podremos escuchar el sonido. Este movimiento se representa gráficamente con una onda oscilante, como observamos en la figura 2.

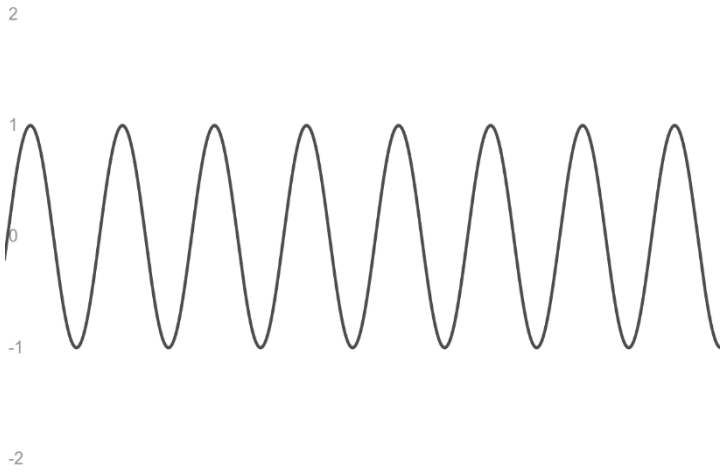

Figura 2. Gráfica de ondas de sonido

Los humanos podemos captar audiblemente un rango de frecuencias que se estandariza entre 20 y 20000 Hz. Los Hertz (Hz) indican la cantidad de ciclos por segundo de una onda. Llamamos ciclo a la trayectoria de la onda cuando vuelve al punto de inicio, en la figura 3 podemos observar cinco ciclos completos.

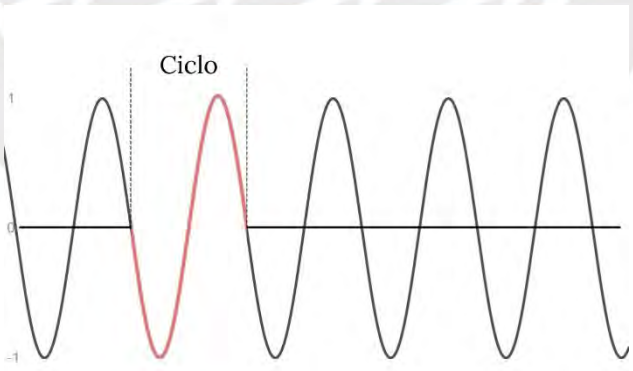

Figura 3. Visualización de ciclos en onda de sonido

La segunda propiedad fundamental es la amplitud, que determina el volumen al que escuchamos un sonido. Gráficamente se determina de acuerdo a qué tan amplia es una onda.

En la figura 4 podemos distinguir entre una onda más amplia que otra, por lo tanto, dicha onda tendrá mayor volumen.

La amplitud se determina por el nivel máximo al que llega una onda (amplitud pico), la amplitud en cualquier otro punto que no sea el pico se denomina amplitud instantánea. Cuando una onda tiene 1 de amplitud significa que la onda oscilará entre su valor positivo y negativo, en este caso, entre 1 y -1; si la onda tiene una amplitud de 0.5, entre 0.5 y -0.5.

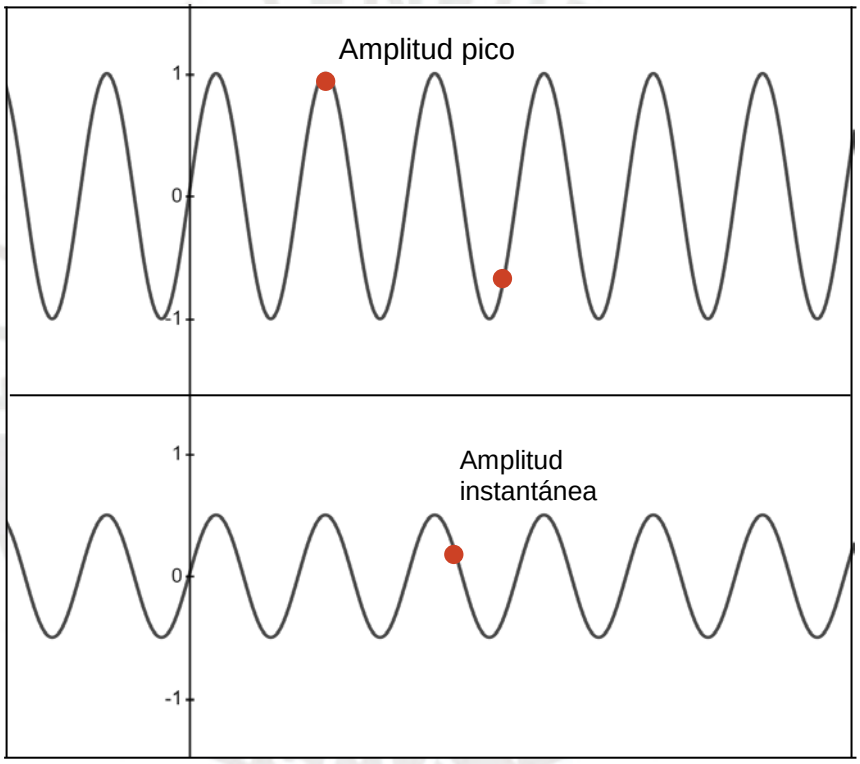

Figura 4. Ondas con amplitud 1 y 0.5

En el estudio de la síntesis de sonido, la amplitud de la suma de ondas con las que se trabaje no debe sobrepasar 1, puesto que es el límite de reproducción de las interfaces de sonido. Si una onda suena a una amplitud de 1.2, ese valor excedente se omitirá y deformará la forma natural de la onda y por lo tanto el sonido sonará distorsionado.

La tercera propiedad es la forma de onda, debido a esta, es posible diferenciar los timbres de los sonidos que se emiten, por ejemplo, podemos diferenciar el sonido de un

trombón y el de una quena por más de que toquen la misma nota. La forma de onda es responsable casi al 100% del timbre de un sonido. En el estudio del sonido desde la síntesis, se usa la onda sinusoidal como la base de las demás ondas. Es a partir de esta que podemos modelar diferentes formas de onda y, con esto, diferentes timbres. Las formas de onda sintetizadas básicas son periódicas.

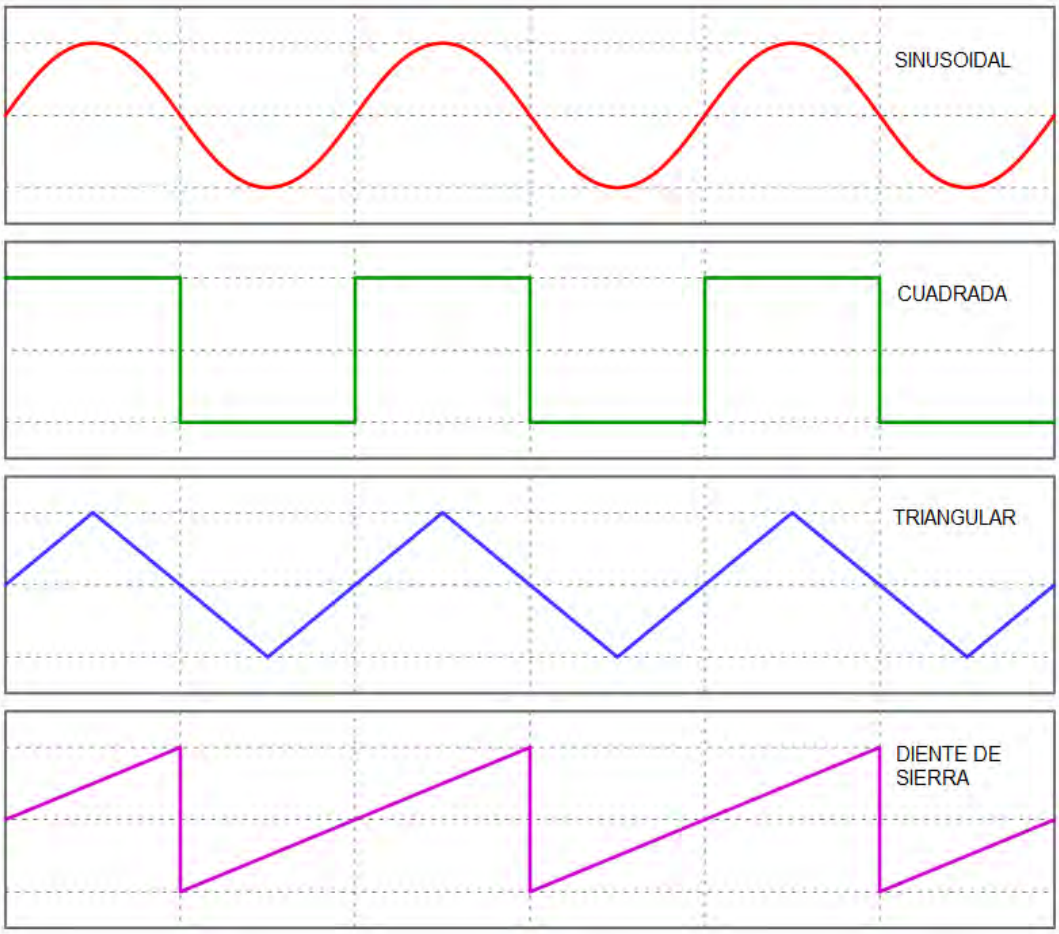

Figura 5. Formas de onda

1.1 SuperCollider: funcionamiento básico del software

El presente proceso creativo propuesto fue desarrollado con diferentes recursos y plataformas. Inicialmente llevé a cabo la exploración utilizando un pedal de *looper*, en donde grababa diferentes sonidos donde después de un cierto número de ciclos grabando capa sobre capa, se pudiera escuchar un conjunto de sonidos al que denominamos ciclos sonoros.

Seguidamente opté por realizar este proceso dentro de un *Digital Audio Workstation* con el uso de un *plugin* de *delay,* al cual le podía asignar un ciclo de hasta 10 segundos de duración. Dentro el DAW o *Digital Audio Workstation* tenía la capacidad de grabar el proceso, lo cual me permitió editar el resultado posteriormente. De esta manera podía mover los sonidos deseados y reordenarlos en distintos lugares del ciclo, podía también eliminar sonidos indeseados, y, además, podía explorar distintos tiempos de duración del *loop*, lo cual ocasionaba que los sonidos se desplacen y el orden de aparición sea distinto. Estos procesos me resultaron muy interesantes debido a que, con pocas modificaciones, el ciclo sonoro generaba un resultado bastante distinto al original y, a partir de estos, podía surgir una propuesta compositiva inesperada. Debo resaltar el gran recurso que significa la exploración de resultados en esta forma de otorgar variación a un proceso creativo fijo. Sin embargo, el uso de estas herramientas digitales me limita a explorar enmarcado dentro un proceso de composición, donde discierno en base a diversas posibilidades dentro de un momento y espacio privado, más no en una situación de ejecución en vivo. Esa fue la razón para haber optado por otros medios que me permitieran ejecutar dichos ciclos sonoros en vivo. Fue en el entorno de programación de sonido SuperCollider en donde pude aplicar los conceptos de los ciclos sonoros en un formato de interpretación en vivo.

Es de mi interés dentro de este capítulo —en donde explicaré los procedimientos que realicé para abordar mi investigación— retratar la forma de trabajo en SuperCollider, no en un sentido de enseñar el uso del software, existen una vasta variedad de libros y videos gratuitos que realizan esta tarea y cualquier interesado sacaría más provecho estudiando alguno de dichas fuentes. Mi intención, en cambio, es poder explicar a usuarios no familiarizados con el software la lógica usada para aplicar los conceptos musicales que propongo acerca de los ciclos sonoros. Además, me parece importante reconocer la minuciosidad con la que se debe trabajar dentro

del software para poder precisar el orden de los sonidos y asignar las capacidades de manipulación del software de acuerdo a mi interés y necesidad.

SuperCollider es un entorno aplicado a la síntesis digital de sonido en tiempo real y a la composición algorítmica. Al mismo tiempo, es un entorno de programación que tiene un lenguaje propio. Este lenguaje hace el uso de una extensa librería de objetos predefinidos, los cuales cumplen funciones específicas. Este tipo de entornos de programación se conocen como OOP (*Object Oriented Programming*), que significa que la estructura del programa se organiza en un sistema ramificado con clases que anidan objetos con funciones parecidas entre sí. Por ejemplo, *UGen* es una clase que agrupa a varios objetos que están creados para generar sonido. Por ejemplo el objeto *SinOsc,* que es uno de los objetos más básicos de SuperCollider, es una función matemática que genera ondas de forma sinusoidal. Cuando utilizamos este objeto debemos definir, además, parámetros de frecuencia y amplitud, para completar la mínima información necesaria para describir un sonido específico de acuerdo a sus propiedades físicas. El usuario entonces interactúa con objetos y declara instrucciones específicas para que el programa pueda interpretar la información escrita y convertir dichas instrucciones en, finalmente, sonido audible. La forma de inserción de código cumple un ordenamiento de programación computacional, con un orden lógico de envío de señal, que se ve transformada por los diferentes objetos con los que interactúa.

En SuperCollider, los objetos que generan ondas de sonido, como *SinOsc,* representan funciones matemáticas cuyos valores fluctúan dentro de un rango de -1 y 1 y, posteriormente esta información se interpreta por los altavoces para generar sonido. Para especificar funciones dentro del software debemos emplear el uso de llaves { }, como podemos observar en la figura 6. Asimismo, el objeto de onda sinusoidal *SinOsc* tiene especificados los parámetros a 150 Hz de frecuencia y 0.2 de amplitud (mul). Finalmente, podemos oír la onda usando el método *play* y ejecutando el código escrito. (en programación, un método representa una instrucción

específica a ciertos valores u objetos, en este caso el método *play* envía la información al servidor de audio, el cual reproduce la onda audio).

```
(
{
    SinOsc.ar(freq: 150, mul: 0.2).dup
}.play
)
```

Figura 6. Instrucción de reproducción de Onda sinusoidal escrita en SC

Esta es la forma más simple de generación de sonidos, en la cual indicamos al software reproducir una onda sinusoidal; sin embargo, en este ejemplo, el sonido no se puede modificar ni controlar una vez ejecutado el código. Este sonido emitido se almacena en el *buffer* del software y suena indefinidamente. Para detenerlo, se necesita aplicar una especie de *killswitch* que apaga todos los sonidos que suenan dentro del software, más, en este proceso, no se está controlando específicamente la onda que hemos diseñado. En el caso de que tengamos dos ondas diseñadas de la manera expuesta reproduciéndose en el software al mismo tiempo, no podemos apagar solamente una o dejar la otra sonando con el *killswitch*. Necesitamos entonces, hacer nuestro código más específico.

Para poder controlar un sonido después de ejecutar su código es necesario asignarle una variable, una palabra asignada por nosotros que puede almacenar información dentro de ella, con la que podamos referirnos a él posteriormente. Como observamos en la figura 7, a la misma función que utilizamos antes le estamos asignando la variable *f*.

```
11 (
12 f = {
13     SinOsc.ar(freq: 150, mul: 0.2).dup
14 }
15 )
16
17 x = f.play
18
19 x.free
20
```

Figura 7. Asignación de variable a función

De este modo, dicha función ahora se alberga en la variable *f*, por lo tanto podemos declarar la expresión *f.play*. Seguidamente, debemos asignar dicha expresión a otra variable, en este caso será *x*. De esta manera, podemos reproducir el sonido al ejecutar la línea de código 17, y lo detenemos ejecutando la línea 19. En otro ejemplo, si nosotros queremos crear un sonido con una duración específica necesitaremos añadir a nuestro código un objeto que cree un envolvente para que controle la amplitud de la onda en el tiempo, en este caso, usaremos un envolvente *ADSR*. En la figura 8 vemos el envolvente que queremos agregar a la función tomada como ejemplo. Tiene un tiempo de ataque de 0.2, en donde llega al punto de amplitud máxima (1), luego decae hasta un 0.8 de amplitud, en 0.2 segundos, y tiene un tiempo de liberación de 2 segundos.

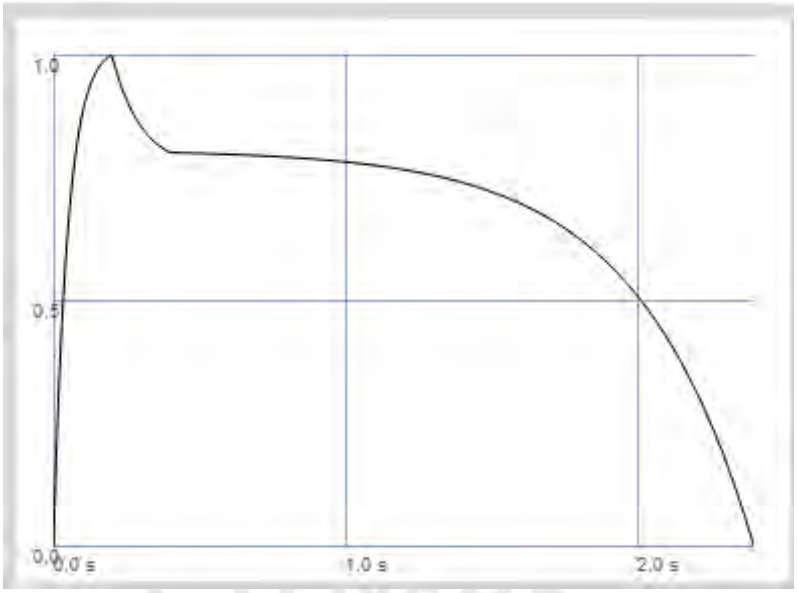

Figura 8. Envolvente ADSR

Para añadir este envolvente a nuestra función debemos construir un algoritmo más complejo en donde operemos los valores de salida de la onda sinusoidal con los valores de la envolvente propuesta. En la figura 9, podemos observar cómo se realiza esta declaración. En primer lugar, se debe entender que un envolvente de tipo *ADSR (attack, decay, sustain, release)* simula el actuar de un teclado al pulsar una tecla, por lo que, al emitirse, el volumen subirá hasta el punto máximo de amplitud, luego decaerá un poco y se sostendrá hasta que se deje de pulsar la tecla. Por esta razón, nosotros declaramos un argumento *(arg)* llamado *gate*, que se operará posteriormente para indicar la acción de dejar de pulsar una tecla, lo que liberará el sonido en un lapso de 2 segundos. Luego, declaramos la variable *env* que encapsulará, en este caso, la información de la envolvente. El objeto *EnvGen* es el generador de envolventes, que recoge la información definida de *Env.adsr,* cuyos valores de salida se operarán con los valores de la salida de la función sinusoidal por medio de una multiplicación, resultando en un sonido con una envolvente. Seguido de declarar el algoritmo, podemos ejecutar la línea 38 para reproducir el sonido. Cuando queramos acabar con el sonido podemos ejecutar la línea 40, lo que modificará el valor del argumento *gate* a 0 y el envolvente se liberará de manera progresiva en el lapso designado.

```
28
29 (
30 f = {arg gate = 1; var env;
31
32     env = Env.adsr(0.2,0.2,0.8,2, curve: [-4,-2,5,]);
33     env = EnvGen.ar(env, gate,doneAction: 2);
34     SinOsc.ar(freq: 150, mul: 0.2).dup * env;
35 }
36 )
37
38 x = f.play
39
40 x.set(\gate, 0)
41
```

Figura 9. Incorporación de un envolvente *ADSR*

Como se puede entender, el proceso de trabajo dentro de este software exige que cada acción que se desee realizar debe especificarse dentro del código, solo de esta manera, el programa ejecutará el sonido tal cual lo necesitemos. Estos pasos explicados muestran un sentido lógico y eficiente de trabajo para los usuarios de SuperCollider, permiten mantener el bloque de código ordenado y claro.

La propuesta de los ciclos sonoros aplicada en SuperCollider consiste en la generación de ondas de sonido con diferentes alturas y dispuestas dentro de un ciclo de una determinada duración. Con una intención parecida a la creación apoyada en pedales de *looper* o *delay*. Busco generar sonidos que cumplan con dos principios: que surjan en momentos aleatorios sin tomar en consideración un ritmo preconcebido, y que la altura de las notas se enmarque dentro de una escala; sin embargo, que no se preconcibe una dirección melódica al momento de crear los ciclos. La funcionalidad de la aplicación de ciclos sonoros tiene que ver con las diferentes formas de organizar los elementos sonoros que componen el ciclo, su orden de aparición, su transformación en el tiempo, el desplazamiento de los elementos entre ellos y otras posibilidades de otorgar variación a la organización de eventos sonoros gracias a la construcción algorítmica. Al enfocar este trabajo dentro de un marco de organización

secuencial de sonidos y no tanto con la generación de timbres por medio de técnicas de síntesis de audio, su aplicación se ubica en el campo de la composición algorítmica.

Para realizar este proceso creativo en un entorno de programación es necesario llevar a cabo 3 procesos fundamentales, son procesos específicos de síntesis y programación que difieren de una construcción básica de una onda de sonido y proponen una manera diferente de enfocar la construcción de instrumentos virtuales. Dichos procesos involucran el uso del principio de la amplitud modulada; la aplicación de iteraciones de procesos computacionales y la expansión multicanal.

1.2. Amplitud Modulada: principio algorítmico de los ciclos sonoros

La amplitud modulada es una técnica de modulación que consiste en la operación de dos ondas, la onda base o *carrier*, y la onda moduladora. Se realiza una operación de multiplicación de valores entre estas dos ondas, modificando la forma de la amplitud. Dentro de mi exploración utilicé la técnica de amplitud modulada de diferentes formas para poder construir instrumentos digitales. Su uso se erige como el principio que otorga la naturaleza de generación de los ciclos sonoros con resultados únicos sobre la manipulación del sonido que la diferencian de otras formas de secuenciar sonidos. Después de un ejemplo sencillo planteo extender un poco más el entendimiento de esta técnica para visibilizar su utilización como pilar dentro de esta investigación.

Como primer ejemplo, usamos una onda sinusoidal de 20 Hz como *carrier*. La otra onda corresponde a la **onda moduladora**, la cual modificará (de acuerdo a su forma de onda) la amplitud de la primera. Este principio se puede entender gráficamente si nos fijamos en la figura 10, en donde, si multiplicamos ambas ondas, obtendremos el resultado de la derecha, que es una onda oscilante cuya amplitud osclia 2 veces por segundo de manera periódica, debido a la **onda moduladora** utilizada.

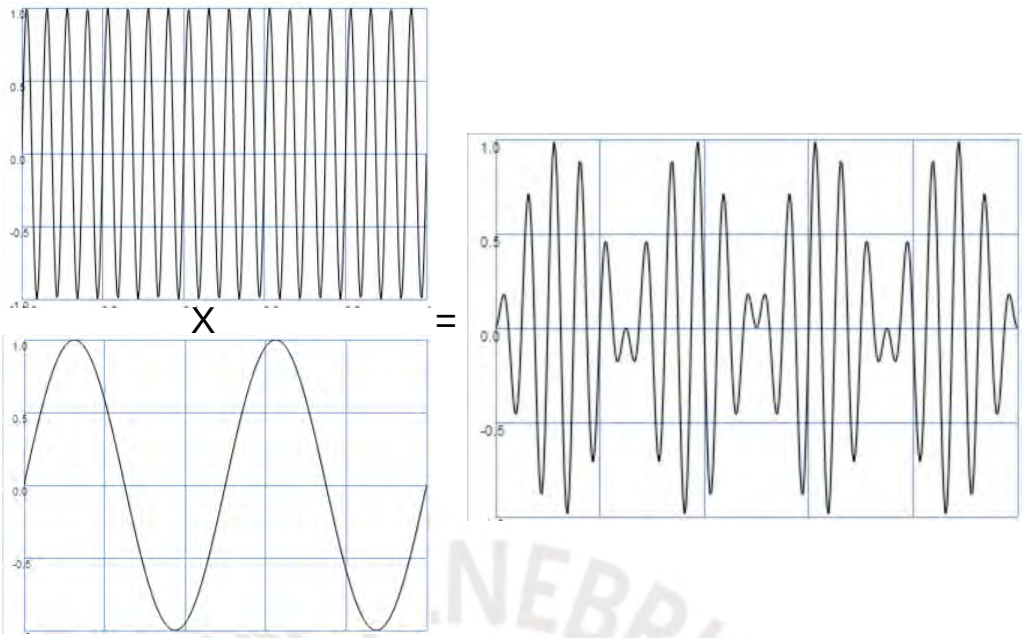

Figura 10. Amplitud modulada

La onda moduladora del ejemplo consiste en una onda sinusoidal con una frecuencia de 2 Hz, es decir, el ciclo de la modulación dura un total 2 segundos. Se puede escuchar el resultado pulsando en los hipervínculos.

Onda moduladora a 2 Hz

Nótese la diferencia en usar diferentes frecuencias para la onda moduladora:

Onda moduladora a 10 Hz

Onda moduladora a 0.2 Hz

La diferencia recae en el tiempo que dura la modulación, no en el sonido base o *carrier*. Cuando la onda moduladora tiene más frecuencia, se genera un efecto semejante a un trémolo, cuando la frecuencia es más baja, se percibe de manera más clara la modulación de acuerdo a la forma de la onda utilizada. En este caso usamos una onda sinusoidal por lo que la modulación de la amplitud es gradual. Si es que utilizamos una onda cuadrada como onda moduladora el sonido se modificará de manera más drástica, sin un *fade* que gradúe el volumen.

Onda cuadrada moduladora a 2 Hz

Como se puede percibir, los valores de amplitud varían solamente entre 1 (volumen máximo) y 0 (silencio) inmediatamente. En este caso, la onda cuadrada (figura 11) se mantiene un 50% del tiempo en 1 y el otro 50% en 0. Esta característica es modificable, se conoce como *duty cycle*, y permite que los tiempos de la amplitud modulada sean diferentes aun cuando la onda está en la misma frecuencia.

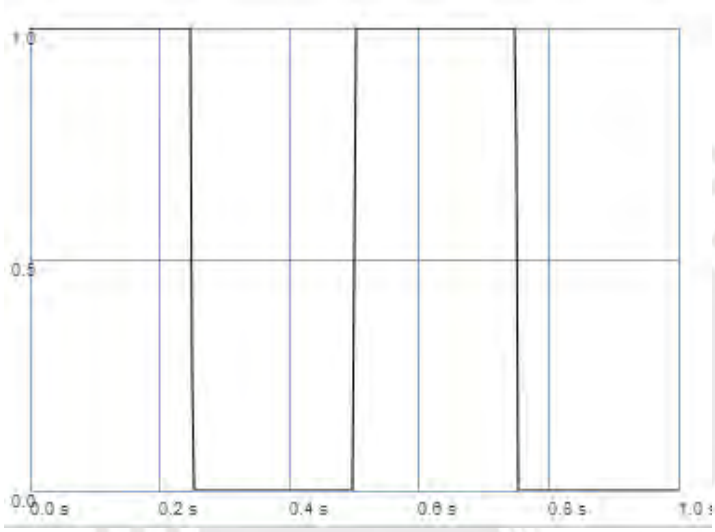

Figura 11. Onda cuadrada de 2 Hz

En la figura 12, se muestra una onda con un *duty cycle* diferente. Cambié su valor, de 0.5 a 0.1. Esto modifica el tiempo en el que la onda estará en valor 1 y valor 0 dentro de un ciclo. En este caso, el valor 1 surge un 10% del tiempo total del ciclo, y el 90% restante, en 0. El resultado consiste en una onda con un pequeño momento con sonido frente a un silencio 9 veces más largo.

Onda cuadrada moduladora a 0.5Hz con un *duty cycle* de 0.1

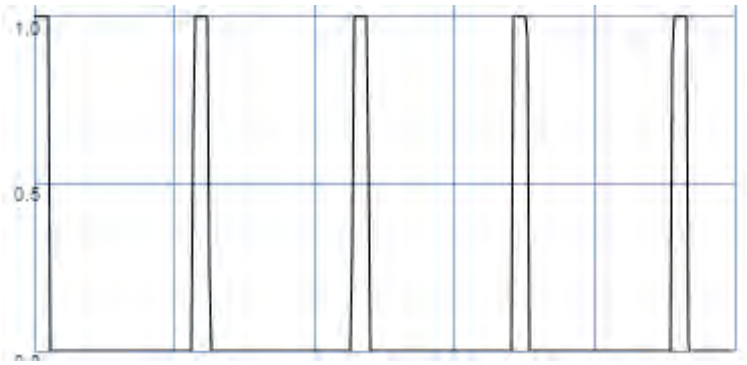

Figura 12. Onda cuadrada con un duty cycle de 0.1

Es de esta manera que un oscilador generará un sonido por un determinado momento y se mantendrá en silencio un tiempo mucho mayor. ¿Qué sucede si repito esta acción de crear un nuevo sonido con diferentes notas cada una con una onda moduladora? En principio, se estarían generando diversos sonidos que surgen cada cierto momento como si se tratara de una secuencia de notas, con lo que podríamos obtener como resultado un ciclo sonoro. Sin embargo, para realizar esta acción de manera más eficiente, podemos programar una instrucción para automatizar este proceso de repetir manualmente la acción de generar una nueva onda de sonido en diferentes momentos. Para esto debemos tomar en cuenta la onda moduladora. Si es que ejecutamos 10 sonidos sin modificar la onda moduladora de amplitud, todos los sonidos sonarían y se quedarían en silencio en el mismo momento. Para solucionar esto, es necesario

que las ondas moduladoras inicien, para cada nota, en distintos lugares, para que así los sonidos se repartan en distintos espacios dentro del ciclo.

Lo que podemos hacer es interactuar con la fase de las ondas moduladoras usadas. La fase es un parámetro propio de las diferentes ondas que podemos utilizar. Su valor indica en qué momento del ciclo una onda inicia su transcurso. Como podemos observar en la figura 13, cambiar el valor de la fase de una onda cuadrada hace que se desplace el momento en el que está en 1 y 0. Con esta lógica, si nosotros desplazamos las ondas moduladoras de nuestros 10 sonidos, estos surgirán en distintos momentos, de acuerdo a la fase otorgada.

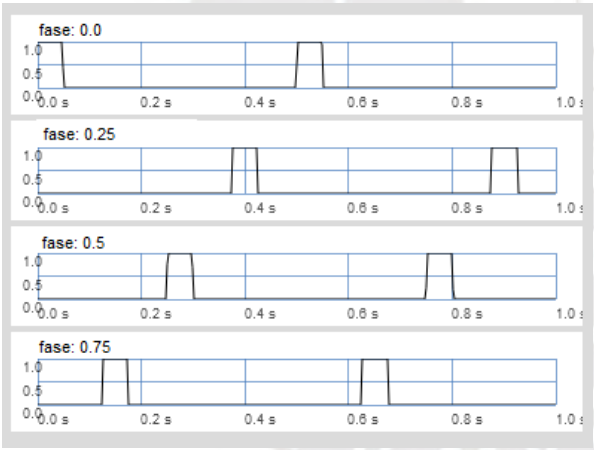

Figure 13. onda cuadrada con distintas fases

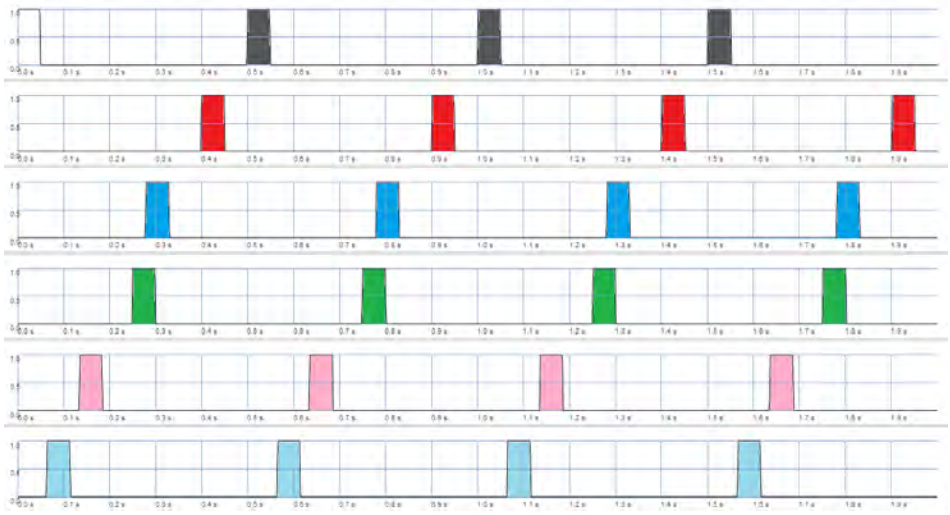

Figura 14. Ondas moduladoras con fases diferentes

En la figura 14 observamos 6 ondas moduladoras controlando la amplitud de 6 osciladores con distintas fases, lo que genera que el sonido de cada oscilador surja en distintos momentos. En la figura 15 se muestra la manera escribir estas instrucciones en el programa. Para comandar, dentro del programa, la instrucción de generar varios osciladores, cada uno con una onda moduladora distinta y cada una con diferentes fases, es necesario recurrir a la iteración del proceso, es decir, repetir un proceso una determinada cantidad de veces en una sola ejecución. En SuperCollider podemos recurrir a diversos métodos para asignar los valores a las fases. Se podría realizar manualmente, o con algún algoritmo, o simplemente ser aleatorio. Para el fin de esta explicación utilizaré una asignación aleatoria. El objeto rrand (range random) que pide 2 argumentos: un valor mínimo y un valor máximo, nos devuelve un número aleatorio entre estos 2 números. La fase de una onda se indica en valores de 0.0 a 1.0. En la figura 14 se muestra el código de creación y reproducción de este algoritmo. En la línea 86 utilizo el objeto rrand(0,1.0) y, cada vez que el proceso se ejecute, cada nota tendrá una fase distinta asignada aleatoriamente.

```
(
f = {
    |freq 330, amp 0.2, pan 0, fase 0|

    var sig,mul;

    mul = LFPulse.kr(0.2,iphase: fase , width: 0.1);
    sig = SinOsc.ar(freq)*mul;
    sig = Pan2.ar(sig, pan, amp);

};
)
x = f.play

(
    a = [];
12.collect({
    a = a.add(f.play(args: [
        \freq, [60,62,64,65,67,69,71,72,76].choose.midicps,
        \fase, rrand(0.0, 1)]))
})
)

a.collect({arg n ; n.release(1)})

```

Figura 15. Algoritmo de iteración

Este principio de uso de ondas moduladoras con diferentes fases es la base funcional de los ciclos sonoros. Inicialmente, el resultado puede parecerse al de un secuenciador convencional. Pero la aproximación es distinta, debido a que, las fases no han sido subdivididas simétricamente como normalmente funciona un secuenciador, sino fueron asignadas aleatoriamente. Además, y esto representa el punto de partida del estudio, las fases pueden modificarse una vez ejecutadas, lo que permite reordenar el surgimiento de los sonidos. Cada onda moduladora puede tener una frecuencia distinta al resto, por lo que cada nota puede tener una duración distinta con el transcurrir del tiempo, entonces la melodía que se forma se transformaría, desplazando la ubicación de las notas poco a poco. De estas maneras la aproximación del uso de una construcción algorítmica de ciclos sonoros se asemeja más al concepto de composición que Brian Eno uso en el disco *Ambient 1: Music for Airports*, que al de un secuenciador, aunque podría imitar a uno si se le programa de tal forma.

La frecuencia de cada onda también es importante, pues en el resultado busco oír sonidos o notas distintas. Lo primero que hago al crear el algoritmo es declarar los argumentos que necesitaré para posteriormente modificarlos en la iteración, en este caso, *freq* representa a

las frecuencias que utilizaré. En la figura 15, en la línea 85 expreso una lista de valores: [60,62,64,65,67,69,71,72,76] (que representan notas *midi* que forman una escala mayor desde Do4) en donde en cada iteración, se elegirá, de manera aleatoria, un valor de esta lista.

Capítulo 2: Construcción de instrumentos digitales en SuperCollider

Los objetivos que he buscado alcanzar con estos estudios han sido poder conseguir una capacidad más grande de manipulación de los elementos que conforman los propuestos ciclos sonoros al aplicar variación a más parámetros que solamente a las fases de las ondas moduladoras. En cada uno, se exponen características únicas de control sobre ciertos parámetros. Las variaciones exploradas en cada estudio, demuestran las ventajas de control sobre el sonido que son posibles gracias a la capacidad de control a través de la programación. Como vimos anteriormente, los valores de los parámetros pueden ser asignados de maneras aleatorias enmarcadas dentro de una extensión específica. En cada uno de los casos, me aproximo a la asignación de valores de una manera distinta. Se emplean diferentes objetos del *software* para su construcción, por lo que la información algorítmica es única. La realización de cada caso me ha permitido discriminar entre los métodos empleados para la construcción del algoritmo y, con esto pude determinar la mejor aproximación a la creación de ciclos sonoros, considerando su versatilidad de manipulación en vivo y peso de procesamiento de la computadora.

De esta manera, el compositor o intérprete puede usar SuperCollider para ejecutar los ciclos sonoros en vivo, en donde puede modificar los parámetros del código para así variar ciertas características sonoras del ciclo.

Los ciclos sonoros generados en SuperCollider son, en términos técnicos, algoritmos y funciones matemáticas que reciben información específica y construyen diferentes sonidos cuyos parámetros se recogen a partir de la introducción de valores numéricos aleatorios. La aleatoriedad juega un papel fundamental en la realización de esta propuesta, pues de esta manera, me puedo alejar de tener una intención preconcebida sobre la línea melódica que se planea componer. Sin embargo, la aleatoriedad está cercada dentro de un margen de

opciones, un rango de valores que está dentro de una concepción musical definida, como podrían ser los grados de una escala, una octava específica, duración, timbre, entre otros.

SuperCollider es un entorno que interpreta instrucciones que han sido escritas por el usuario, es por esto que, como constructor del código, necesito ser muy específico en cuanto a cómo quiero que el programa se comporte. Cada capacidad de control que quiera otorgarle, como la posibilidad de cambiar la duración del ciclo, al *patch* en SuperCollider tiene que ser diseñada y codificada por mí. Dentro de esto, he tenido que optar por diferentes caminos de aproximación para encontrar soluciones para los fines que me propuse. Los estudios realizados en este capítulo se presentan de acuerdo al nivel de complejidad de construcción del código y de la cantidad de procesos algorítmicos que se requieren para que el *patch* realice procesos con la posibilidad de control sobre los parámetros.

Para todos los casos, el oscilador generador de la señal audible siempre será una onda sinusoidal, con el fin de poder evaluar de manera equitativa la cantidad de procesamiento que genera cada *patch*. La onda sinusoidal genera una sola frecuencia sin armónicos, por lo que su utilización representa la señal más sencilla de generar en términos de proceso computacional. Por otro lado, al tener como objetivo principal la búsqueda de formas de manipulación del orden de aparición de sonidos, su recurrencia, frecuencia y amplitud a través de algoritmos y del uso de la aleatoriedad, el timbre o "color" de los sonidos no es un factor determinante en esta investigación. Por esta razón, el uso de ondas sinusoidales representa la aplicación más básica en cuanto a la elección del timbre del instrumento, frente a una posibilidad infinida de timbres que se pueden adecuar en el código.

Se debe entender que, los sonidos se generan con diferentes osciladores, en donde cada uno tiene un sonido asignado que se emite cada cierto tiempo y se mantiene así hasta que, en algunos casos, se le indique cambiar sus parámetros por otros. El empleo de varios osciladores en programación de sonido se conoce como expansión multicanal, donde cada

canal está asignado a un oscilador independiente. En los dos primeros estudios, emplearé una expansión multicanal de 10 osciladores, es decir que, en cada ciclo sonoro, habrán hasta 10 sonidos independientes ejecutándose simultáneamente. En el último caso, emplearé un proceso de iteración 10 veces, que realizará el proceso de generación de sonido 10 veces.

2.1. Caso 1: Frecuencias ordenadas simétricamente

El primer caso consiste, en términos sencillos, en la construcción de un algoritmo que genera ondas sinusoidales cuyas frecuencias se encuentran afinadas en intervalos simétricos determinados por el intérprete. La duración del ciclo, así como el *sustain* de las notas, la frecuencia base, los intervalos y el volumen se pueden modificar en tiempo real.

```
(
SynthDef(\est1, {
    |amp 0.2, cycle 0.3, width 0.1, base 45,step 2|
    var sig, phases, muls, freqs;

    phases = 10.collect{Array.interpolation(12, 0.0, 1).choose.round(0.05)};

    muls = (VarSaw.kr(freq: cycle, iphase: phases, width: width)**3).clip;

    freqs = Array.series(10, base,step).midicps;
    freqs = freqs * [0.99,1,1.02].choose;

    sig = SinOsc.ar(freq: freqs, mul: muls);

    sig = sig * amp;

    sig = sig.collect({
        arg n;
        Pan2.ar(n, rrand(-1,1.0))});

    sig = sig.sum;

    Out.ar(0, sig)
}).add
)
```

Figura 16. Codigo del estudio 1

En la figura 16 se puede observar el código que empleé para la construcción de este instrumento digital. Con el código expreso el funcionamiento del ciclo sonoro, es ahí donde yo debo indicar específicamente cuáles son las instrucciones para producir sonido. Este proceso se realiza combinando el funcionamiento de distintos objetos de SuperCollider y generando una comunicación matemática entre ellos. Los objetos propios del software están

resaltados en color celeste y el resto de información se reparte entre argumentos de los objetos, métodos y valores numéricos.

En este caso, tenemos un *SynthDef*, con el que defino el instrumento virtual al que llamo *est1*, este instrumento tiene los argumentos siguientes: *amp* (volumen), *cycle* (duración del ciclo), *width* (*sustain* de las notas), *base* (nota midi base) y *step* (intervalos simétricos). Los argumentos representan valores numéricos que yo puedo colocar en el lugar que desee, los utilizo para poder cambiar posteriormente el valor que representan inicialmente.

Seguidamente, declaro las variables: *sig* (señal de sonido), *phases* (fases), *muls* (control de amplitud), *freqs* (frecuencias). En la variable *muls* indico la forma de manipulación de la amplitud de la onda, con esto controlo la recurrencia de su aparición dentro del ciclo. Para la onda moduladora de amplitud utilizo el objeto *VarSaw,* la cual genera una onda triangular (figura 17). Este objeto tiene 3 argumentos que debo especificar (*freq, iphase, width). Freq* corresponde a la recurrencia de la onda triangular, para indicar este valor uso el argumento declarado *cycle*, que corresponde a 0.3, que indica que cada 0.333 segundos se repetirá la onda moduladora de amplitud. El argumento *width* corresponde al *duty cycle* de la onda triangular, modificando su valor predeterminado de 0.5, nos permite movilizar el pico de la onda hacia la izquierda o derecha. Seguidamente elevamos al cubo la salida de la onda triangular y luego utilizamos el método *clip*. Con esto podemos perfilar los valores de la onda para que se manipule el volumen de manera más controlada (figura 18).

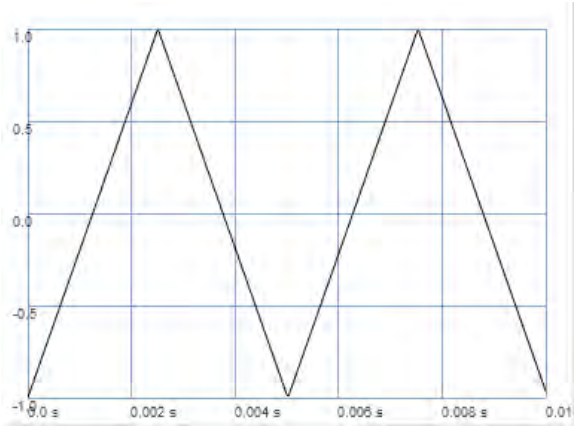

Figura 17. Onda triangular

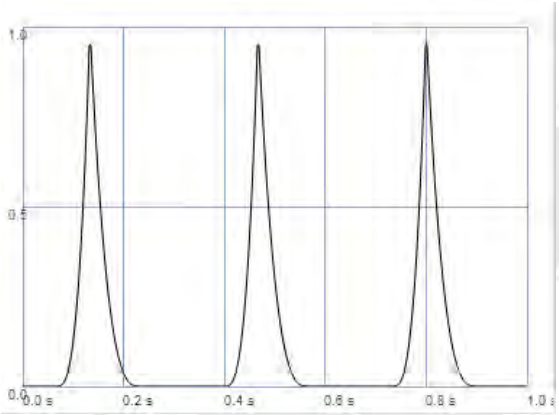

Figura 18. Onda triangular con duty cycle 0.4 y elevada al cubo

El argumento restante del objeto *VarSaw* es *iphase*, este es el objeto con el que controlamos la fase inicial de la onda. En este argumento inserto la información proveniente de la variable *phases*, cuyo valor es el resultado de elegir un número aleatorio de una lista de 12 valores simétricos entre 0 y 1 ([0.0, 0.1, 0.2, 0.25, 0.35, 0.45, 0.55, 0.65, 0.75, 0.8, 0.9, 1.0]), 10 veces. Esta segunda lista resulta en, por ejemplo [0.25, 0.65, 0.25, 0.0, 0.8, 1.0, 0.8, 0.1, 0.25, 1.0] con la que señalaré la fase inicial de cada oscilador. Entonces los 10 osciladores tendrán fases iniciales distintas. Es decir que, tendrán diferentes órdenes de existencia dentro del ciclo, y su aparición puntillista formará ritmos en vez de que todos surjan al mismo tiempo como un *cluster*.

La variable *freqs* representa también una lista de valores numéricos. Estos, indicarán la frecuencia de los osciladores. Para este caso, he optado por ordenar las listas de frecuencias en intervalos numéricos simétricos utilizando el objeto *Array.series* que requiere los argumentos "número de elementos, inicio, *step*". En el *patch*, estos son: 10, base (45), step (2) lo que crea la lista: "[45, 47, 49, 51, 53, 55, 57, 59, 61, 63]". Estos números de la lista representan notas midi, que se ordenan en números enteros, cada uno representa un semitono, por lo que cada 12 números se cumple una octava. Por convenio, la nota 45 representa la nota La2 (frecuencia de 110 Hz) y 63, Re#4 (frecuencia de 311.13). Posteriormente, a este resultado se multiplica con un valor aleatorio entre 0.99 y 1.02 para desafinar muy ligeramente las frecuencias.

2.2. Caso 2: Multichannel con modificación del ritmo interno al cambiar las fases de cada elemento

En este caso, he buscado modificar los valores de la fase de los elementos, con el fin de otorgar ritmos diferentes entre los elementos que conforman el ciclo sonoro. Esto se consigue gracias al principio de la amplitud modulada. La forma de onda que controla la amplitud de la señal inicia, para cada canal (*multichannel*), en un lugar diferente de su ciclo. Decido usar una forma de onda cuadrada para manipular la amplitud, con la posibilidad de controlar su *duty cycle*. Sin embargo, hay un problema, y es que el objeto *LFPulse*, con el que se crean las ondas cuadradas, no admite posibilidad de modificar su fase una vez iniciada la onda, de hecho, su argumento se llama *iphase* (fase inicial). He podido solucionar este problema al declarar una afirmación Booleana. Observemos en la figura 19 la construcción del instrumento digital en código.

```
(
SynthDef(\est2, {
    |amp 0.2, t_trig 1, cycle 0.3, step 0.09, midi 45, lthan 0.4|
    var sig, phases, freqs, muls;

    phases = Array.series(10,0.0,step);

    muls = SinOsc.kr(cycle, phases*2pi) > lthan;

    freqs = Demand.kr(t_trig, 0, Drand([0,3,7,12,18,19],inf)!12);

    sig = SinOsc.ar(midi.midicps * freqs.midiratio, phases, muls);

    sig = sig.collect({arg n;
        Pan2.ar(n,rrand(-1,1.0))});

    sig = sig.sum * amp;

    Out.ar(0, sig);
}).add
)
```

Figura 19. Código de estudio 2

El código consiste, otra vez, en un SynthDef, al que llamaremos ahora *est2.* Contiene los argumentos: *amp* (amplitud), *t_trig* (disparador), *cycle* (ciclo), *step* (el valor de avance de una serie de números), *midi* (el número de la nota midi), *lthan* (símbolo de menor que).

En la primera variable, *phases*, declaro una lista de 10 elementos del tipo *series*, que ya se observó en el anterior caso. Esta vez, los elementos de lista se van a asignar a la fase de la onda cuadrada que modula la amplitud, asignándose en cada canal una fase distinta. La lista empieza en 0.0 y aumenta sus valores de 0.09 en 0.09 de acuerdo al valor del argumento *step*. En este caso, la lista es [0.0, 0.09, 0.18, 0.27, 0.36, 0.45, 0.54, 0.63, 0.72, 0.81], Considerando que el rango total de la fase va entre 0 y 1, con esta lista abarcamos un 81% de la fase, y el restante 19% se reservará siempre como un espacio de silencio. En la siguiente variable *muls*, hago el uso de un *SinOsc* seguido del símbolo y argumento > *lthan*. El signo > declara una afirmación Booleana que devuelve un 1 si la afirmación es correcta, y un 0 si la afirmación no lo es. Mientras el valor de salida de un *SinOsc* sea mayor a 0.4, el resultado es 1, si el SinOsc es menor a 0.4, el resultado se transforma en 0. Por lo que, como observamos

en la figura 20, los valores de salida de un SinOsc oscilan entre -1 y 1, por lo que más del 75% esta declaración es falsa. Esta operación resulta en una onda cuadrada, como observamos en la figura 21. Este resultado se genera tras la operación booleana del mayor que.

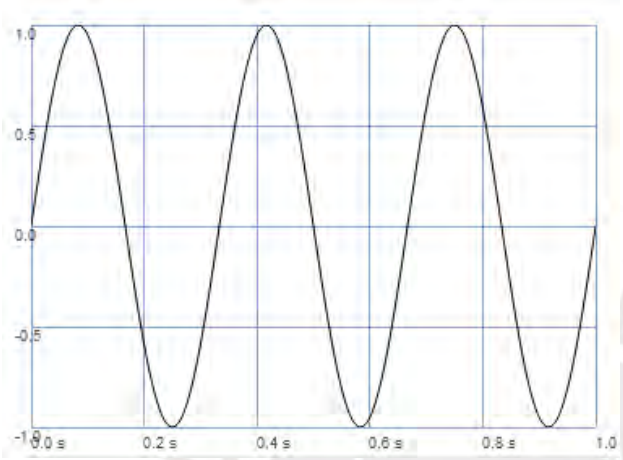

Figura 20. Onda sinusoidal, sus valores son menores a 0.4 la mayor parte del ciclo

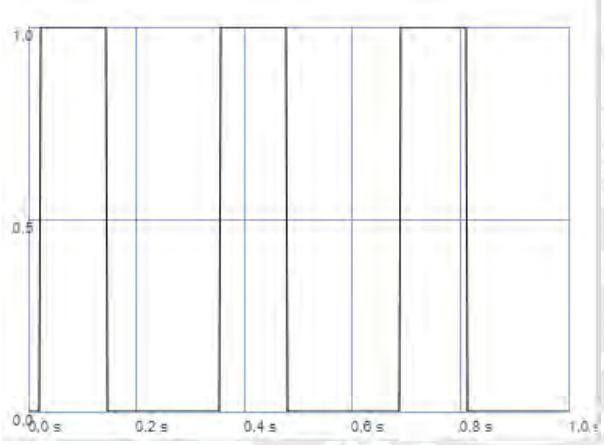

Figura 21. Onda sinusoidal operada con una operación Booleana

Seguido a esto, tenemos la variable *freqs*, que requiere el uso del objeto *Demand,* con el que se puede solicitar elementos que se reciben de una lista, en este caso, del objeto *Drand*, que selecciona valores de manera aleatoria de la lista de valores [0,3,7,12,18,19] que, al interpretarlos como semitonos, representan un acorde menor con un tritono. Al ejecutar el código, un elemento, por ejemplo, el *3* se selecciona y se utiliza como valor actual que determinará la frecuencia del oscilador.

A continuación, insertamos las anteriores variables dentro de los parámetros de un *SinOsc*, que será nuestra señal de audio. Como frecuencia del *SinOsc* uso el valor del argumento *midi*, posteriormente dicho valor se transforma en frecuencia con el método *midicps* (*midi to cicles per second)*. Seguidamente operamos con la lista de valores de *freqs* y las convertimos en una ratio de intervalos, en este caso, indico un acorde menor con un tritono. Seguidamente, en un proceso de iteración, asigno los valores de paneo de cada señal de sonido de manera aleatoria y finalmente multiplico la señal por una amplitud baja de 0.2 para controlar los niveles de volumen.

Este *SynthDef* debe ser asignado a una variable para luego modificar sus parámetros.

2.3 Caso 3: Empleo de iteración en un instrumento

En el afán de obtener una capacidad de manipulación mayor buscando objetos cada vez más complejos y específicos, me di cuenta de que la respuesta podía alojarse en una operación más sencilla, utilizando diferentes resultados aleatorios en los argumentos asignados a la señal. Para lograr esto, ya no opto por generar señales *multichannel* desde la construcción del SynthDef, sino, la lógica consiste en construir un *SynthDef* que inicie con argumentos aleatorios, y con esto, iterar 10 veces dicho *SynthDef's*, en donde cada uno inicie con argumentos diferentes. Gracias a esta disposición de osciladores, puedo solicitar un valor numérico diferente para cada argumento de cada oscilador (*SynthDef*), con lo que se puede generar aún más variación dentro de todo el instrumento.

En este caso, son importantes dos secciones del código: la construcción del instrumento en código (Figura 22) y el comportamiento de la reproducción del mismo (Figura 24). Con la primera, diseñamos el instrumento virtual, con la segunda, lo ejecutamos 10 veces, cada vez con valores diferentes en sus argumentos.

```
(
SynthDef(\est3, {
    |amp 0.2, midi 42, cycle 0.15, cycleRatio 1, phases 0, mix 0.2, room 0.2, damp 0.5, pan 0, st 0, pow 100|

    var sig, muls;

    muls = SinOsc.kr(cycle*cycleRatio, phase: phases).range(0,1)**pow;

    sig = SinOsc.ar((midi + st).midicps);

    sig = sig * amp * muls;
    sig = Pan2.ar(sig, pan);
    sig = FreeVerb.ar(sig, mix, room, damp);

    Out.ar(0,sig);
}).add
)
```

Figura 22. Código de estudio 3

Este *SynthDef* tiene una construcción más sencilla en cuanto al uso de objeto e interconexión entre ellos. Aquí hago uso de una mayor cantidad de argumentos pues voy a necesitarlos para modificar el comportamiento del instrumento y además utilizo un reverb al final para darle efecto de ubicación al sonido. Tenemos los argumentos: *amp* (volumen), *midi* (frecuencia en notas midi), cycle (ciclo), cycleRatio (ratio del ciclo), phases (fase), mix (mix del reverb), room (tamaño del cuarto), damp (filtro de agudos), pan (posición panorámica), st (semitonos), pow (potencia). La primera variable que declaro es *muls*, con la que designo la forma de onda que manipula la amplitud, la frecuencia de esta onda se conforma por dos argumentos *cycle* y *cycleRatio* que se multiplican. La fase se otorga con el argumento phases, que en este caso inicia con 0. Seguidamente utilizo el método *range* para escalar la salida de la señal entre 0 y 1, de esta manera controlaremos con más precisión la amplitud. Finalmente elevo este output a la potencia 100 para poder afilar la forma de una onda sinusoidal. En la figura 23 se observa el resultado en la forma de onda después de elevar una señal a sinusoidal al ciento.

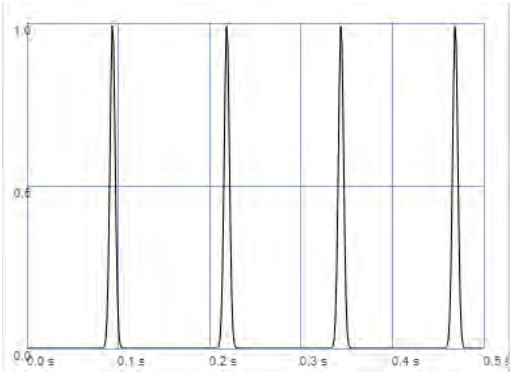

Figura 23. Onda sinusoidal elevada a 100

La siguiente variable es la señal de audio, la cual será una onda sinusoidal cuyas frecuencias se representan con la suma de los argumentos *midi* y *st.* En este caso, *midi* representa la nota midi y *st*, los intervalos en semitonos a partir de la nota midi base. La suma entre estos dos valores por fin se transforma en ciclos por segundo. Finalmente, se asigna un valor de paneo a la señal.

El segundo bloque del código, graficado en la figura 24, indica la reproducción del código. El presente ciclo sonoro se ejecutará dentro de una lista, en donde sus elementos corresponden a un *Synth* diferente. El código previamente declarado reproduce una sola señal, es decir, una sola nota específica, determinada aleatoriamente como podemos escuchar en este ejemplo. En el ejemplo escuchamos una frecuencia que se repite cada cierto tiempo, la altura y el espacio en el que se encuentra dentro del ciclo se ha determinado aleatoriamente de acuerdo al algoritmo de selección de un elemento de la lista 60 + [0,2,5,7,8,9,12] para la frecuencia, dichos valores son notas midi que posteriormente se escalan en valores en Hertz. Y para la fase se escoge un número aleatorio entre el 0 y el 6.28 (2pi) Para crear los ciclos sonoros, debo reproducir este algoritmo varias veces, cada vez, los valores de altura, fase y paneo, se sortearán entre posibilidades aleatorias. Para el proceso de utilización de listas, es importante primero declarar una lista en blanco a la que llamaremos *~est3*. Una vez creada la lista, ya puedo operar en ella y asignarle elementos que serán diferentes osciladores. El

bloque de código que empieza en la línea 19 y está entre paréntesis es la instrucción de generación de osciladores. Primero se declara el enunciado *~est3 = ~est3.add.* Este enunciado indica que cada vez que se ejecute el código, se añadirá un elemento nuevo en la lista. Posteriormente especifico el tipo de elemento que incrustaré en la lista. En este caso indico que deseo reproducir el *Synth ~est3*, y justo en esta parte, designo que los valores iniciales de los argumentos *phases, freq, st,* que antes eran estáticos, ahora sean elegidos aleatoriamente. Cada vez que ejecute el código estoy generando un elemento nuevo dentro de la lista. Este proceso lo realizo 10 veces y obtengo una lista de 10 elementos en donde cada uno representa a un *synth* activo. En este caso, indicamos que, para cada ejecución de los *synths*, la fase (*phase*) se escoja aleatoriamente entre 0 y 6.28 (pi), la frecuencia inicia la nota midi 60, para todos los casos. Finalmente, los semitonos (*st*) se escoge un número de aleatorio de la lista [0,2,5,7,8,9,12].

```
19 (
20 ~est3 = [];
21 10.collect({
22         ~est3= ~est3.add(
23             Synth(\est3,
24                 [
25                 \phases, rrand(0.0,2pi),
26                 \midi, 60,
27                 \st, [0,2,5,7,8,9,12].choose,
28                 \pow, 10;
29                 ]
30         ))
31 })
32 )
```

Figura 24. Reproducción del instrumento ~est3

Una vez realizada la iteración, nosotros podemos recurrir al mismo método para modificar los valores de sus argumentos. La gran ventaja de trabajar con diferentes *synths* es puedo asignar valores diferentes para los argumentos del instrumento. En este sentido, puedo modificar la frecuencia y los ritmos de manera aleatoria dentro de un rango de trabajo. Realizaré el primer ejemplo de modificación de argumentos sobre las frecuencias de los

synths. En la imagen 2.8 podemos ver que la construcción del codigo es parecida a la anterior: Se iteran 10 acciones, en donde cada una representa a la modificación individual de cada elemento de la lista de *synths*, en vez de escoger elementos de la lista [0,2,5,7,8,9,12], ahora los elementos se escogen entre [0,2,3,7,8,10,12]. Y cada vez que se ejecute este bloque de código, las frecuencias de los osciladores cambiarán por otros parámetros aleatorios. 10.collect indica la cantidad de iteraciones, seguidamente abrimos paréntesis y declaramos que la función tiene un argumento *n*, que representa al número de la iteración. Se cuenta desde cero por lo que, si hicimos 10 iteraciones tenemos 10 elementos, del 0 al 9. Seguidamente hacemos referencia a la lista que acabamos de crear, *~est3* y declaramos que ~est3[n] donde n es cada uno de los 10 osciladores que están generando sonido con diferentes atributos. A cada oscilador, le indicamos con .set que cambiará los valores de *\st*, el argumento de semitono, por un elemento escogido aleatoriamente de la lista [0,2,3,7,8,10,12]. Cada vez que se ejecute este bloque de código, las frecuencias de los osciladores cambiarán a notas afinadas en una escala menor sin cuarta.

```
35 //ST
36 (
37 10.collect({ arg n;
38     ~est3[n].set(
39         \st, [0,2,3,7,8,10,12].choose,
40         )
41 })
42 );
```

Figura 25. Código de modificación de valores

Esto lo podemos aplicar a cualquier otro argumento:

```
//GRUPO DE ARGUMENTOS
(
10.collect({ arg n;
    ~est3[n].set(
        \st, [0,2,4,7,9,10,12,19,22].choose,
        \cycle, (1.4*[0.15,0.13,0.16]).choose,
        \midi, [36,60].choose,
        \pow, 200,
        \pan, rrand(-1,1.0),
        )
})
);
```

Figura 26. Modificación de varios argumentos en una sola ejecución

2.4 Resultados

El uso y exploración de los 3 instrumentos que se presentan en los estudios de caso revelan una serie de ventajas de manipulación sobre los elementos que conforman los ciclos sonoros en comparación al uso de un instrumento acústico, sintetizador modular, *drum machine* o secuenciador. Esto es posible gracias a las naturalezas matemáticas propias de un entorno de programación como lo es SuperCollider. Si bien cada uno tiene capacidades de manipulación diferentes y desde diferentes aproximaciones, los parámetros que podemos modificar esbozan sobre la frecuencia, ritmo y duración de cada sonido sintetizado. Cada caso constituye un instrumento independiente que genera ciclos sonoros con características de construcción específicas. De acuerdo a su construcción, se albergan una variedad de parámetros manipulables únicos, por lo que en este capítulo se abordarán los resultados de la exploración en los instrumentos creados y se detallarán los resultados en base a los siguientes conceptos:

Capacidad de manipulación:

Frecuencia (*st, midi, freq*)

Amplitud (*amp*)

Ritmo (*phase*)

Duración:

De los elementos sonoros (*duty cycle, pow*)

Del ciclo (*cycle*)

El desarrollo de las conclusiones se acompañará de un video explicativo en donde se observará la introducción de nuevos valores numéricos que modifican los parámetros mencionados, además, el resultado es más entendible.

2.4.1 Instrumento de estudio 1 (~est1)

El instrumento 1 es un generador de ciclos sonoros muy sencillo, crea lienzo que tiene una duración universal asignada al argumento *cycle*. El ritmo se designa solamente al ejecutar el instrumento, es decir, ya no se puede modificar. La duración correspondiente a los elementos se designa con el argumento *width*, que va entre 0.0 y 1. La amplitud se designa con el argumento *amp*. Y finalmente, la frecuencia, que es la que más capacidad de variación tiene se designa con un objeto que genera una lista de números de manera específica. Esta lista genera números que ascienden desde un número base en intervalos de números enteros, estos se asignan en los argumentos *base* y *step*.

El ciclo sonoro de este primer ejemplo se constituye de una secuencia de sonidos cuyas alturas se encuentran separadas interválicamente por valores microtonales. En este caso quiero una lista de diez elementos que empiece en el número 50 y suba de 3 en 3: *Array.series(10,50,3).* El step inicial determina una separación interválica por 3 semitonos, es decir, terceras menores, que forma una escala disminuida. La lista resultante es, entonces: [50, 53, 56, 59, 62, 65, 68, 71, 74, 77]. Sin embargo, al modificar el valor del *step* a 3.1 genero un ordenamiento distinto de frecuencias, que en este caso podemos ya considerar como microtonal, pues el intervalo sería de 3.1 semitonos. Lo que genera que las frecuencias se ordenan en intervalos de 3.1 tonos. Siendo la lista ahora [50.0, 53.1, 56.2, 59.3, 62.4, 65.5,

68.6, 71.7, 74.8, 77.9]. Esta aproximación desde ya nos devuelve un resultado que no se puede replicar en un instrumento acústico debido a los ordenamientos escalares de las alturas propuestas. Posteriormente se van incrementando los valores del *step* por décimas hasta llegar a 4, una separación de tercera mayor, que genera una escala aumentada.

Frecuencia: \base = 50, \step = 3,

Amplitud: \amp = 0.2,

Ritmo: \phases = lista de 10 elementos con valores aleatorios entre 0.0 y 1

Duración: \width = 0.2 (figura 27)

Primer resultado del estudio 1 (video)

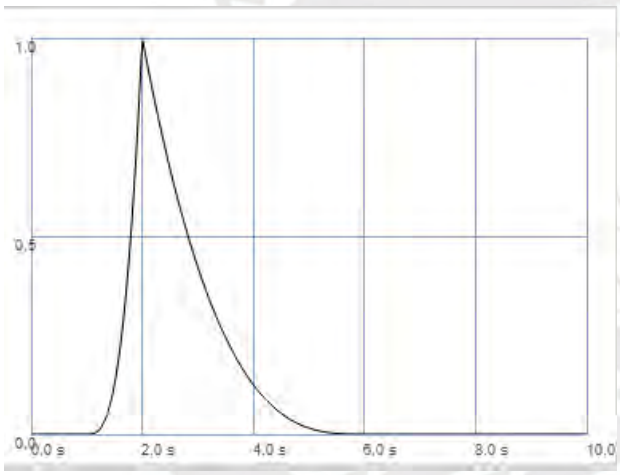

Figura 27. Duración de las señales con un duty cycle de 0.2

En el siguiente ejemplo propongo un ciclo con frecuencias muy cercanas entre sí que generen vibraciones. Para esto designo la lista de frecuencias como una que empieza en 55 y avanza cada 0.06 valores, lo que me otorga finalmente la lista [50.0, 50.06, 50.12, 50.18, 50.24, 50.3, 50.36, 50.42, 50.48, 50.54]. estas frecuencias surgen de manera aleatoria de acuerdo a la selección de la lista en *phases*. El resultado sonoro consiste en un ciclo sonoro que genera un efecto de trémolo a distintas velocidades debido al fenómeno de *beating,* que surge al reproducir dos o más frecuencias cercanas, sus ondas se cancelan por el poco desfase

de frecuencia que hay (figura 28) y generan un efecto de vibración de la intensidad. En el ejemplo audiovisual 2 se puede apreciar mejor el cambio sonoro y la manipulación del software.

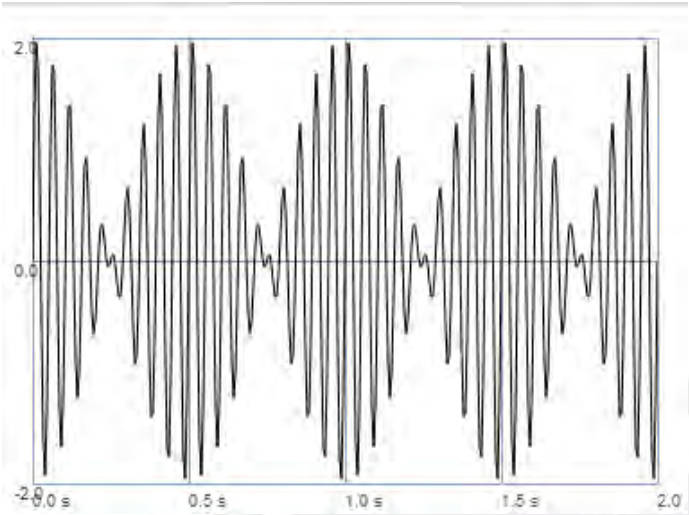

Figura 28. Efecto producido por dos ondas con frecuencias cercanas

Este resultado sonoro es, otra vez completamente posible gracias a la naturaleza matemática del instrumento, con capacidad de manejar frecuencias muy cercanas entre sí. Además de los resultados rítmicos que se generan, la armonía generada a corta distancia brinda sonidos que podrían bien utilizarse como un único elemento dentro de una composición.

Segundo resultado del estudio 1 (video)

2.4.2 Instrumento de estudio 2 (~est2)

El instrumento del estudio 2 es un generador de ciclos sonoros con ciertas variaciones. La duración del ciclo se cambia con el argumento *\cycle*. Las frecuencias que le otorgamos ya no tienen que ser simétricas, podemos elegir una lista de frecuencias o notas midi para que se escojan de manera aleatoria. Una vez iniciado el instrumento, la lista de frecuencias no puede cambiar; sin embargo, podemos pedir a la computadora que se vuelvan a escoger aleatoriamente frecuencias de la lista creada. Este cambio de frecuencias se ejecuta con el argumento *\t_trig*. Para modificar el ritmo, se controlan las fases de las ondas. Los valores se

designan generando una lista de números que asciende de manera simétrica. De la misma forma que en el ejemplo anterior. Controlamos el intervalo de crecimiento de la lista para cambiar los ritmos de los elementos del ciclo sonoro. La duración de los elementos se genera especificando el valor de *\lthan* y va entre 0.0 y 1, mientras más cerca este a 1 la onda cuadrada será más ancha en 0. La amplitud se declara con el argumento *\amp.*

En este ejemplo, voy a explorar las capacidades tímbricas del instrumento en un ciclo sonoro de 2.5 segundos cuyas notas dibujan un arpegio de menor: [la2, do3, mi3, la3, re#4 y mi4]. Los sonidos generados dibujan una línea melódica que fue designada de manera aleatoria. En este caso, busco obtener diferentes timbres modificando la duración de los elementos individuales. Para este caso tengo que modificar los valores de *lthan*. Mientras más cerca esté a el sonido será más corto (figura 29). En el video 3 podemos observar el comportamiento del instrumento, al incrementar la anchura de la onda cuadrada, los sonidos tendrán una duración más corta. Cuando le asignamos a *lthan* un valor de 0.999 los sonidos son tan cortos que su timbre se asemeja más a un instrumento de percusión. Modificar el argumento *cycle* hará que la duración del ciclo sea menor y la secuencia ocurra más rápido.

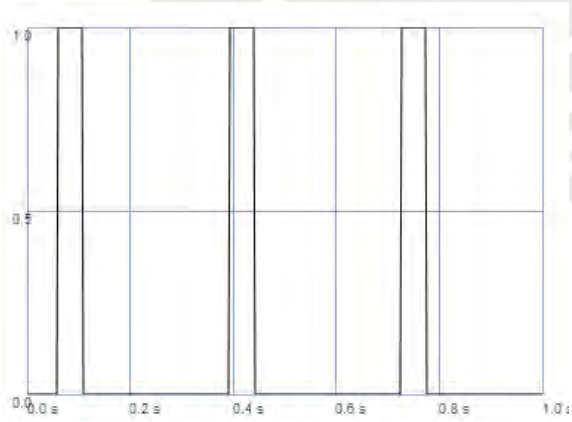

Figura 29. lthan 0.9: la onda cuadrada es más ancha en 0 y angosta en 1

Primer resultado del estudio 2 (video)

La siguiente exploración del instrumento de estudio 2 consiste en la variación de las notas del ciclo sonoro. La lista de valores que modifica la frecuencia se genera de la siguiente manera: en este caso, se declara la lista [0,5,8,12,15,20]. Al tener como nota midi base 50, las posibles notas en este ciclo sonoro serán: [50, 55, 58, 62, 65, 70]. El presente instrumento exhibe una característica peculiar, al ejecutar el argumento *t_trig* con valor 1 los valores de la lista que controla las frecuencias del ciclo se vuelven a escoger aleatoriamente. Cada reorganización devolverá un nuevo orden de los elemento contenidos en la lista, lo que generará una nueva dirección melódica en la secuencia, hecha por la elección aleatoria de los elementos de la lista. Esta lista se puede actualizar, es decir, sus elementos se pueden volver a escoger aleatoriamente si es que ejecutamos de nuevo el argumento *t_trig* con el valor 1. En el ejemplo audiovisual 4 se pueden observar las diferentes líneas melódicas que surgen.

Segundo resultado del estudio 2 (video)

Como podemos apreciar en los ejemplos audiovisuales, podemos manipular varios parámetros de los ciclos sonoros. La modificación de estos parámetros hace posible que las frecuencias elegidas, velocidades de reproducción, duración de elementos puedan transformar la naturaleza inicial del instrumento digital, convirtiéndolo en un secuenciador de notas y ritmos, pero con la característica de un ciclo sonoro, un espacio con notas controladas a través de la amplitud modulada.

2.4.3 Instrumento de estudio 3 (~est3)

El instrumento del estudio 3 consiste en un generador de ciclos sonoros más complejo que los anteriores dos, pero más versátil debido a que la expansión se realiza con un proceso de iteración. Los valores números que se asignan a un argumento pueden ser diferentes en

cada uno de sus elementos, que son escogidos de una lista aleatoria que el intérprete puede modificar en tiempo real. La **frecuencia** se puede modificar en conjunto con los argumentos *midi* y *st*. La **amplitud**, con el argumento *amp*. El **ritmo** de los elementos se designa con el argumento *phases*, en este caso, sí podemos modificar en tiempo real los valores de *phases*. La **duración** de los elementos se asigna con el valor *pow*. La duración del ciclo, con *cycle*.

Propongo crear un ciclo sonoro con una **duración** de 6 segundos, para esto le asigno el valor 0.166 al argumento *cycle*. Las **frecuencias** se determinan con los argumentos *midi* y *st.* En este caso la nota base es la nota midi 60 correspondiente a un Do4, y sus intervalos se otorgan en la lista *st*: [0,4,5,7,9,12,14,21]. El **ritmo** se genera de manera aleatoria, en este caso con valores aleatorios de una interpolación entre el 0 y 6.2832 (2pi), que corresponden a los valores de la fase de una onda sinusoidal. La **duración de los elementos sonoros** se especifica con el argumento *pow*. El **volumen**, con el argumento *amp.*

En el ejemplo audiovisual modifico todos los argumentos que el instrumento contiene, con el fin de exponer la versatilidad que este instrumento digital nos permite.

Resultado del estudio 3 (video)

Los ciclos sonoros se constituyen de elementos sonoros repartidos en un espacio. Estos elementos cumplen con instrucciones o principios de existencia como altura, amplitud, duración y timbre determinados, y sus valores se asignan a través de algoritmos en SuperCollider. La naturaleza de ordenamiento matemático del software brinda resultados de manipulación no solamente versátiles, sino expande las posibilidades acústicas del fenómeno sonoro. Por ejemplo, en la armonía. La frecuencia es determinada numéricamente en notas midi, organizada en 12 semitonos. Por ejemplo, una nota 60 equivale a Do4, 61 a Do#4, 64 a Mi4 y así... Al estar en un entorno de programación, podemos usar números decimales, y de esta manera, generar intervalos mucho menores a un semitono, lo cual brinda resultados

armónicos diferentes. En otro ejemplo, el ritmo de los elementos del ciclo se puede desplazar de manera simétrica al modificar las fases de los elementos que componen el ciclo con un objeto que genere una serie de números que asciende simétricamente como el objeto *Array.series*. Asimismo, podemos generar un efecto de polirritmia entre los elementos sonoros si a cada uno le asignamos una duración diferente en donde el ciclo sonoro ya no tiene una duración fija que se repite, sino esto dependerá del mínimo común múltiplo de las duraciones de los elementos.

```
(
~est3 = [];
10.collect({
        ~est3= ~est3.add(
            Synth(\est3,
                [
                \midi, 50,
                \st, [0,4.2,7,11.5,16,19].choose,
                \cycle, [0.15,0.2,0.25,0.3].choose,
                \pow, 1000,
                \amp, 0.3,
                ]
        ))
})
)
```

Figura 30. Creación de polirritmias con ciclos diferentes

Conclusiones

Los resultados de estos estudios de caso reflejan un desarrollo continuo sobre la construcción de un instrumento virtual que genera ciclos sonoros. El desarrollo recae en la búsqueda de la eficiencia de un código escrito, que cumpla su objetivo de la mejor manera, esto es, resolver los problemas planteados por el usuario con mayor eficacia. En la presente investigación, el objetivo fue claro: construir un instrumento que pueda manipular los elementos de un ciclo sonoro de maneras que no puedo hacer con otros medios, como al usar un pedal *looper* o un Digital Audio Workstation. Entonces, el plan de acción en la programación del instrumento fue el de, continuamente, incorporar más posibilidades de manipulación al instrumento digital. Por esta razón, cada caso de estudio ejemplifica opciones de manipulación diferentes de acuerdo a su construcción, aún con el mismo principio en mente. Estas diferentes perspectivas scon adaptaciones de esquemas de trabajo que yo, como constructor del código diseño. Como programador, estoy cercado dentro del conocimiento que tengo del software, del empleo del pensamiento algorítmico, y la lógica matemática. En este sentido, el proceso normal de creación de un programa (en este caso el programa es el instrumento virtual) aborda muchos más esquemas de trabajo, los cuales son deliberadamente descartados o actualizados a una versión más eficiente. Los ejemplos, además de ilustrar perspectivas compositivas diferentes, retratan etapas diferentes de mi propia investigación, las cuales se pueden entender como una evolución de los instrumentos construidos.

En cuestiones de estructura, el primer estudio de caso, el instrumento 1, genera ciclos sonoros cuyas frecuencias están separadas simétricamente y que además podemos modificar esta separación en tiempo real otorgando valores numéricos diferentes. Lo mismo no sucede con la manipulación de las fases de las ondas moduladoras. Si bien, se determinan aleatoriamente al momento de declarar el código, no es posible modificar dichos valores en

tiempo real. En contraste, el instrumento 2 trata de expandir las cualidades del primero. Aunque la separación simétrica de las notas del ciclo sonoro genera resultados sonoros interesantes, esta naturaleza se puede considerar aún limitante pues no se puede agregar más variedad. Esta versión incorpora en su estructura la posibilidad de escoger una lista de notas seleccionadas de antemano, lo que permite al usuario escoger frecuencias a elección, como una escala menor, como en el ejemplo. Asimismo, las fases de las ondas moduladoras se pueden modificar en tiempo real alterando el valor de *step*. De esta forma, el segundo instrumento permite más capacidad de manipulación en comparación al primero; sin embargo, se pueden seguir añadiendo más posibilidades de control que busquen resolver problemas o limitantes que se pueden encontrar. El instrumento 3 representa el código más eficiente de los casos expuestos, este funciona utilizando otro principio de programación conocido como iteración. Este permite modificar los parámetros de frecuencia y fase de la onda moduladora de amplitud, así como los valores de las alturas de los sonidos de manera individual, es decir, permite otorgar un valor distinto para cada oscilador que exista dentro del ciclo sonoro, logrando una capacidad exponencialmente más variada que los dos anteriores ejemplos. Curiosamente, además, el código escrito es más corto que los dos ejemplos anteriores, esto puede ser un resultado del azar de mi investigación tanto como un efecto inevitable. De todas maneras, esto demuestra que un código programado siempre puede optimizarse.

Si bien hay una propuesta de crear un generador de ciclos sonoros a través de la programación, como se puede presumir, no hay un punto final exacto en el que el instrumento se haya finalizado de construir. Las ciencias de la computación ya han mostrado estar en un constante desarrollo imparable. Por lo que es natural que este código siga evolucionando y mejorando en cada revisión futura. Esto no quiere decir que el desarrollo esté aún incompleto. Considero que los ejemplos utilizados ejemplifican la vasta versatilidad que los instrumentos

permiten, donde, además, cada uno de ellos pueden otorgar una infinidad de posibilidades sonoras. Esto en un principio es posible gracias a la naturaleza digital del entorno que se utilizó para construir los instrumentos: SuperCollider, a diferencia de un sintetizador modular, que está sujeto a la capacidad de osciladores que los módulos tienen, en el mundo digital, ese límite está sujeto solamente a la capacidad de procesamiento de la computadora que se usa. Es decir, si el procesador lo permite, puede reproducir 1, 2, 30 o 300 osciladores solamente comandando tal número en el programa. Por otra parte, la capacidad de asignar valores numéricos a los parámetros de los objetos usados, genera un resultado con una exactitud matemática imposible para otras herramientas de creación sonora, como un instrumento acústico.

Así como podemos utilizar microtonos con una exactitud decimal en los instrumentos construidos debido a la naturaleza digital del entorno, también podemos manipular la organización de los sonidos en el tiempo o *ritmo* con un pensamiento matemático y algorítmico a través del control de las ondas moduladoras de amplitud. Es en esta área específica en que la capacidad ilimitada de la programación y el fundamento conceptual de creación de los ciclos sonoros se combinan y ejecutan un comportamiento peculiar que marca una diferencia a este estudio con respecto a otras herramientas de secuenciación de sonido, tanto en módulos físicos, como en un sintetizador como en un entorno digital, como lenguajes de secuenciación como TidalCycles, FoxDot o la clase de objetos de secuencia *patterns*, de SuperCollider, generalmente utilizados para secuenciar sonidos. Este comportamiento peculiar se genera por la utilización de la amplitud modulada, específicamente, la utilización una onda moduladora individual para cada oscilador que se utilice. Es decir, que, si creamos un ciclo sonoro con 15 osciladores, existen 15 ondas moduladoras, una para cada oscilador. El instrumento de estudio 3 nos permite organizar cada variable asignada. En este caso, las ondas moduladoras tienen 3 argumentos que puedo modificar en tiempo real: *cycle,*

cycleRatio, phase. Los dos primeros controlan la frecuencia del ciclo y el tercero la fase de la onda. Los valores que se otorguen a los parámetros, además, pueden ser diferentes para cada oscilador. Siguiendo con el ejemplo, como los valores controlan la duración de los elementos en un ciclo, al modificar las 15 ondas moduladoras con valores diferentes, se empezarán a generar desfases y desplazamientos condicionados a sus frecuencias. De igual manera, al modificar los valores de la fase con diferentes valores, los sonidos se pueden reorganizar dentro de la secuencia generada.

Finalmente, podemos decir que los instrumentos digitales que generan ciclos sonoros resultan ser completamente viables para una ejecución en vivo. Brindan al ejecutante o intérprete una capacidad de manipulación además de infinita, debido a la naturaleza de la utilización del código, única en cuanto su aplicación se estructura desde la capacidad de manipular individualmente ondas moduladoras de amplitud, donde su modificación puede generar secuencias que se desplazan y desfasan entre sí.

Referencias bibliográficas

Bagés, J. (2011). Sobre las nuevas tecnologías musicales y los sistemas musicales i nteractivos. *Espacio Sonoro* (23), 1-25. Recuperado de http://www.joanbages.com/arxius-web/articles/NuevasTecnologias.pdf

Barleet, B. & Ellis, C. (2009). Music autoethnographies: Making autoethnography sing/Making music personal*. Australian Academic Press*. Bartleet, B. L & Ellys C. Recuperado el 27 de mayo del 2021 de https://www.researchgate.net/publication/44960958_Music_autoethnographies_Making_autoethnography_singMaking_music_personal

Berenguer, Xr. (2002). Arte y tecnología: una frontera que se desmorona. *Artnodes: revista de arte, ciencia y tecnología* (2), 6-10. Recuperado de https://es.scribd.com/document/362146548/Arte-y-Tecnologia-Berenguer

Cage, J. (1937). *El futuro de la música: credo*. Recuperado de: http://www.ccapitalia.net/reso/articulos/johncage/futuromusica.htm

Cipriani, A. & Giri, M. (2010). *Electronic music and sound design: Theory and practice with Max/MSP vol. 1*. Roma: Contemponet s.a.s.

Collins, N. & Mclean, A. & Rohrhuber, J. & Ward, A. (2003). Live coding in laptop performance. *Organised Sound, 8*, 321-329. doi: 10.1017/S135577180300030X.

Eno, B (1979). Pro Session: The Studio as Compositional Tool. *Down Beat 50*. Recuperado en http://music.hyperreal.org/artists/brian_eno/interviews/downbeat79.htm

Fieldsteel, E. (2012). Eli Fieldsteel SuperCollider tutorials & compositions. Recuperado de https://www.youtube.com/c/elifieldsteel

Gutiérrez, C. (2014). *Laptop performance y software libre. Caso práctico: programación en SuperCollider, instrumento virtual beat.* (Tesis de maestría, Facultad de Música, UNAM, Ciudad de México, México).

Keislar, D. (2009). A historical view of computer music technology. *The Oxford Handbook of Computer Music*, pp. 11-43. Oxford University Press.

Lysaker, John. (2017). Turning Listening Inside Out: Brian Eno's Ambient 1: Music for Airports. *The Journal of Speculative Philosophy, 31* (1), 155-176. https://doi.org/10.5325/jspecphil.31.1.0155

Lobato, J. (2021). *Jardines alquímicos. Creación asistida por computadoras químicas.* (Tesis de licenciatura, Facultad de música, Universidad Autónoma de México, Ciudad de México. Recuperado de: https://issuu.com/jaime_lobato/docs/tesis_jaimelobato_jardinesalquimicos_final

Magnusson, T. (2014). Herding Cats: Observing live coding in the wild. *Computer Music Journal, 38* (3), 8-16. https://doi.org/10.1162/COMJ_a_00216

McCartney, J. (2002). Rethinking the Computer Music Language: SuperCollider. *Computer Music Journal, 26* (4), 61-68. https://doi.org/10.1162/014892602320991383

O Connor, N. (2018). *Diffusing the Norm - Brian Eno's Music for Airports (1976)*. Divergence Press. 10.5920/DivP.2019.03.

Recarte, O (2021). Recarte Blog Wordpress. Recuperado el 2 de marzo de 2022, de https://recarteblog.wordpress.com/blog/

Reich, S. (2002). *Writings on Music, 1965- 2000*. Oxford: Oxford University Press

Reverb Machine (2019*). Deconstructing Brian Eno's "Music for Airports*. Recuperado de https://reverbmachine.com/blog/deconstructing-brian-eno-music-for-airports/

Rhys, P. (2016). Smart Interfaces for Granular Synthesis of Sound by Fractal Organization. *Computer Music Journal, 40* (3) 58–67. https://doi.org/10.1162/COMJ_a_00374

Rodriguez, J. (2017). *Sistemas algorítmicos en las artes y sus procesos compositivos [relación medio-imagen-cuerpo]. Caso de estudio: Altamisa.* (Tesis de maestría,

Facultad de arquitectura, arte y diseño, Universidad de Guajanato, falta ciudad y país). Recuperado de http://www.repositorio.ugto.mx/handle/20.500.12059/412

Stevenson, B. (1997). *Paintings in sound*. *Vertical time in the ambient music of Brian Eno.* (Tesis de bachillerato, Facultad de Humanidades,Victoria University of Mancheater, Manchester, Inglaterra).

Tabuena, A. (2018). *Electronic and Chance Music*. doi: 10.13140/RG.2.2.23248.23048.

Tamm, E. (1988). *Brian Eno... His music and the vertical color of sound*. Londres: Faber & Faber

Vail, M. (2014). *The Synthetizer: A comprehensive guide to understanding, programming, playing, and recording the ultimate electronic music*. Nueva York: Oxford University Press.

Wesler, P. (2018). *Proceso creativo en la composición musical con tecnología electrónica.* Buenos aires: UNTREF

Woo Park, J. (2021). *Tag Archives: SuperCollider*. Joowonpark.net. Recuperado de: https://joowonpark.net/tag/supercollider/

Xambó, A & Lerch, A. & Freeman, J. (2019). Music Information Retrieval in Live Coding: A Theoretical Framework. *Computer Music Journal, 42* (4) 9–25. https://doi.org/10.1162/comj_a_00484

Xenakis, I. (1963). *Formalized music: thought and mathematics in composition*. Nueva York: Pendragon Press. Recuperado de: https://monoskop.org/images/7/74/Xenakis_Iannis_Formalized_Music_Thought_and_Mathematics_in_Composition.pdf

Anexos

Enlaces a videos:

Primer resultado de estudio 1:
https://drive.google.com/file/d/1NYMkTd7HHSorFf6nQf7DylbDDGvBVfe0/view?usp=sharing

Segundo resultado de estudio 1:
https://drive.google.com/file/d/1jyZ2RMSPJ6vgn0lbXepqoylgGQEeOBQx/view?usp=sharing

Primer resultado de estudio 2:
https://drive.google.com/file/d/1s87oqlcjTOUeOFP0PXqbCEnOrrOx5nMy/view?usp=sharing

Segundo resultado de estudio 2:
https://drive.google.com/file/d/1Vze8KIxt75Kyw88boO_hktdFtW6Bp0CH/view?usp=sharing

Resultado de estudio 3:
https://drive.google.com/file/d/1AA4LEz7i2p6UNGBu8kq8lgMI2--RSkRm/view?usp=sharing

Código en SuperCollider:

```
//CICLOS SONOROS
//3 ESTUDIOS


//////////////////
//        estudio 1        //
//////////////////



//declaración del instrumento
(
SynthDef(\est1, {
        |amp 0.2, cycle 0.3, width 0.1, base 45,step 2|
        var sig, phases, muls, freqs;

        phases = 12.collect{Array.interpolation(10, 0.0, 1).choose.round(0.05)};

        muls = (VarSaw.kr(freq: cycle, iphase: phases, width: width)**3).clip;

        freqs = Array.series(10, base,step).midicps;
        // freqs = freqs * [0.99,1,1.02].choose;
```

```
	sig = SinOsc.ar(freq: freqs, mul: muls);

	sig = sig * amp;

	sig = sig.collect({
		arg n;
		Pan2.ar(n, rrand(-1,1.0))});

	sig = sig.sum;

	Out.ar(0, sig)
}).add
)

//reproducción del instrumento
x = Synth(\est1)

//modificación de parámetros
//cambia los números y aprieta ctrl + Enter

x.set(\width, 0.01);	//0.0 > width < 1.0
x.set(\step, 4.1);		//0.0 > step < 12.0
x.set(\cycle, 0.4);		//0.0 > cycle < 30
x.set (\base, 60);		//40 > base <80




//////////////////
//	estudio 2	//
//////////////////

//declaración del instrumento
(
SynthDef(\est2, {
	|amp 0.1, t_trig 1, cycle 0.3, step 0.09, midi 45, lthan 0.4|
	var sig, phases, freqs, muls;

	phases = Array.series(10,0.0,step);

	muls = SinOsc.kr(cycle, phases*2pi) > lthan;

	freqs = Demand.kr(t_trig, 0, Drand([0,3,7,12,18,19],inf)!12);
```

```
	sig = SinOsc.ar(midi.midicps * freqs.midiratio, phases, muls);

	sig = sig.collect({arg n;
		Pan2.ar(n,rrand(-1,1.0))});

	sig = sig.sum * amp;

	Out.ar(0, sig);
}).add
)

//reproducción del instrumento
x = Synth(\est2)

//modificación de parámetros
x.set(\step, 0.03);
x.set(\step, 0.1);
x.set(\step, 0.27);	// 0.01 > step < 1
x.set(\cycle, 0.7);
x.set(\midi, 43)
x.set(\lthan, 0.995, \midi, 45)
x.set(\lthan, 0.9)
//cada vez que se ejecuta t_trig, las notas se reordenan
x.set(\t_trig, 1)




//////////////////
//	estudio 3	//
//////////////////


//declaración del instrumento
(
SynthDef(\est3, {
	|amp 0.2, midi 42, cycle 0.15, cycleRatio 1, phases 0, mix 0.2, room 0.2, damp 0.5,
pan 0, st 0, pow 100|

	var sig, muls;

	muls = SinOsc.kr(cycle*cycleRatio, phase: phases).range(0,1)**pow;

	sig = SinOsc.ar((midi + st).midicps);
```

```
        sig = sig * amp * muls;
        sig = Pan2.ar(sig, pan);
        sig = FreeVerb.ar(sig, mix, room, damp);

        Out.ar(0,sig);
}).add
)


//reproducción del instrumento
(
~est3 = [];
10.collect({
                ~est3= ~est3.add(
                        Synth(\est3,
                                [
                                \phases, rrand(0.0,2pi),
                                \midi, 60,
                                \st, [0,2,5,7,8,9,12].choose,
                                \pow, 10;
                                ]
                ))
})
)


//modificación de parámetros:
//semitonos:
(
10.collect({ arg n;
        ~est3[n].set(
                //      [0,2,5,7,8,9,12]        //
                \st, [0,2,3,7,8,10,12].choose,
                )
})
);

//GRUPO DE ARGUMENTOS
(
10.collect({ arg n;
        ~est3[n].set(
                \st, [0,2,4,7,9,10,12,19,22].choose,
                \cycle, (1.4*[0.15,0.13,0.16]).choose,
```

```
                \midi, [36,60].choose,
                \pow, 200,
                \pan, rrand(-1,1.0),
                )
})
);


//fase:
(
10.collect({ arg n;
        ~est3[n].set(
                \phases, rrand(0,2pi),
                )
})
);

//freqcuencia base
(
10.collect({ arg n;
        ~est3[n].set(
                \freq, [36,60].choose,
                )
})
);


//paneo
(
10.collect({ arg n;
        ~est3[n].set(
                \pan, rrand(-1,1.0),
                )
})
);


//frecuencia de la onda moduladora
(
10.collect({ arg n;
        ~est3[n].set(
                \cycle, ([0.15,0.13,0.16]).choose
                )
})
```

```
);


//duty cycle de onda moduladora
(
10.collect({ arg n;
        ~est3[n].set(
                // \cycle, [0.3,0.6,0.9].choose
                \pow, 1000                )
})
);
```