UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Programa de Maestría y Doctorado en Música

Escuela Nacional de Música
Centro de Ciencias Aplicadas y Desarrollo Tecnológico
Instituto de Investigaciones Antropológicas

LAPTOP PERFORMANCE Y SOFTWARE LIBRE.
CASO PRÁCTICO: PROGRAMACIÓN EN SUPERCOLLIDER, INSTRUMENTO VIRTUAL BEAT

TESINA APOYADA EN TRABAJO PRÁCTICO
QUE PARA OPTAR POR EL GRADO DE:
MAESTRÍA EN MÚSICA (Tecnología Musical)

PRESENTA:
CARLOS OCTAVIO GUTIÉRREZ LÓPEZ

TUTOR
DR. JORGE RODRIGO SIGAL SEFCHOVICH
Programa de Maestría y Doctorado en Música

MÉXICO, D. F. NOVIEMBRE 2014

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Programa de Maestría y Doctorado en Música

Escuela Nacional de Música
Centro de Ciencias Aplicadas y Desarrollo Tecnológico
Instituto de Investigaciones Antropológicas

LAPTOP PERFORMANCE Y SOFTWARE LIBRE.
CASO PRÁCTICO: PROGRAMACIÓN EN SUPERCOLLIDER, INSTRUMENTO VIRTUAL BEAT

TESINA APOYADA EN TRABAJO PRÁCTICO
QUE PARA OPTAR POR EL GRADO DE:
MAESTRÍA EN MÚSICA (Tecnología Musical)

PRESENTA:
CARLOS OCTAVIO GUTIÉRREZ LÓPEZ

TUTOR
DR. JORGE RODRIGO SIGAL SEFCHOVICH
Centro Mexicano para la Música y las Artes Sonoras (CMMAS)
Director

MÉXICO, D. F. NOVIEMBRE 2014

Agradecimientos

A la Universidad Nacional Autónoma de México y a la Escuela Nacional de Música, en especial al programa de Maestría y Doctorado en Música, directivos, administrativos y maestros, así como a la Unidad Académica de Posgrado por el apoyo otorgado.

Al Dr. Rodrigo Sigal, así como al Centro Mexicano para la Música y las Artes Sonoras (CMMAS), por el apoyo y las facilidades recibidas para la realización del presente trabajo.

A Sergio Luque por la orientación para el trabajo de programación en SuperCollider.

Agradecimientos a título personal.

A Sonia y a Carlos Enrique, cuyo esfuerzo y dedicación cotidiana admiro y utilizo como inspiración para afrontar nuevos retos, siempre.

Ricardo, Erick, Roberto y el pequeño Diego Josep, este trabajo va dedicado a ustedes, sirva acaso de motivación para perseguir sus anhelos, sin importar lo lejanos o difíciles que se presenten, que ya la búsqueda en sí misma es suficientemente gratificante.

/ Pudo representarse su extensión, los agrupamientos simétricos, la gráfica, el valor expresivo; tuvo ante sí esa cosa que ya no es música pura que es dibujo, arquitectura, pensamiento, y que permite recordar la música. /

// Marcel Proust, En busca del tiempo perdido

S.boot;

Índice

Introducción

El auge en la comercialización de las computadoras personales y la consecuente masificación de los sistemas operativos comerciales ha provocado una fuerte asociación entre la idea de que el *software* es generalmente un producto comercial, sin embargo, desde los albores de la informática, las universidades consideran el software como un producto de investigación, esto sumado a la idea de compartir el código fuente de los programas fue lo que principalmente provocó el desarrollo de diversos sistemas informáticos. El software libre y de código abierto es entonces por naturaleza una de las líneas de investigación más interesantes para los programas de investigación relacionados con la creación de herramientas tecnológicas.

La importancia del software libre no es sólo la posibilidad de crear nuevas aplicaciones de libre acceso sino que también propone un modelo colaborativo de producción y distribución de las mismas. La producción colaborativa y el libre acceso al código de un programa han mostrado ya los beneficios que acarrean, con distintos ejemplos durante toda la historia de la ciencia computacional. Por lo que continuar con esta forma de actuar es una obligación académica en la búsqueda por el beneficio social que cualquier universidad tiene como motor de existencia.

En los últimos años este interés por los programas junto a la proliferación de computadoras portátiles ha provocado que una gran cantidad de artistas incursionen en el campo de la programación, lo que lleva a que la práctica de crear software sea (en algunos casos) el trabajo del artista, aprovechando los desarrollos tecnológicos para crear sus propias herramientas. Esta dinámica afecta los procesos tradicionales que la industria alrededor de la música ha establecido, por lo que hoy en día existen artistas sonoros, programadores, compositores, intérpretes, etcétera; que comparten las herramientas y los procesos que utilizan para su creación, lo que da pie a una cultura musical mucho más global, que se nutre de los distintos nodos que la componen. En mi opinión, este proceso lo encuentro lo suficientemente interesante para generar un trabajo que sustente y apoye estas nuevas dinámicas a través del software, la creación colectiva y la libre transmisión del mismo. Pero además, dentro de mi experiencia puedo resumir la importancia que considero tiene la distribución del software libre y de diversos contenidos a través de Internet, ya que esto es lo

que me permitió tener conocimiento de la existencia de una maestría en el campo de la tecnología musical, así como del programa que utilizo para el trabajo: SuperCollider, cuyo uso en las clases y asesorías durante el posgrado, pero que sin la constante colaboración colectiva que se da en las listas de correos del programa, foros, tutoriales y demás recursos en la red, hubiera requerido mucho mayor número de clases y tiempo invertido para generar la parte práctica del trabajo. Se enuncia aquí una propuesta para creación colectiva a través del software libre como algo que el campo de la Tecnología Musical requiere tomar dentro de sus líneas prioritarias de investigación desde las universidades y el marco de lo institucional.

El trabajo consta de cuatro capítulos que describiré brevemente:

El primer capítulo es un acercamiento a la historia de la música por computadora y cómo la tecnología afecta la creación musical, a partir de diversas fuentes se realiza un recorrido cronológico de la música por computadora y algunos antecedentes a partir del inicio del siglo XX, para finalmente, retomar el tema en la actualidad y puntualizar el modo en que las nuevas tecnologías se encuentran con paradigmas establecidos.

En el segundo capítulo se plantea la relación del software libre con el trabajo, se abordan primero los fundamentos del mismo y la forma en la que rápidamente se ha ido desarrollando y sentando las bases para nuevas prácticas tecnológicas, planteando también que se ha llegado a establecer como una manera que contribuye a la creación artística.

El tercer capítulo explica la conexión entre la tecnología musical y el software libre, en este caso un programa específico como es SuperCollider, el cual tiene la peculiaridad de haber sido originalmente un software comercial pero a partir de su inclusión como software libre experimentó un crecimiento muy fuerte en la base de usuarios y por supuesto de desarrolladores que lo han convertido en una de las alternativas más interesantes para acercarse a la creación musical e interpretación en directo desde diferentes perspectivas, una de ellas el denominado *laptop performance* el cuál al entrar en diálogo con paradigmas de la práctica musical emerge como una nueva práctica para tomarse en consideración.

El cuarto capítulo describe la herramienta desarrollada para el trabajo junto a un manual de usuario que se distribuirá de forma libre con la herramienta, explicando la interfaz gráfica y el funcionamiento de los distintos parámetros, dejando las bases para futuras modificaciones que cada usuario requiera hacer para satisfacer sus necesidades.

Finalmente se anexa el código *fuente*, el cuál por supuesto puede ser descargado en línea

junto a las instrucciones para su correcta ejecución, cuyo sitio para descarga se específica al inicio del anexo. Es evidente que en una aplicación de esta naturaleza, el código fuente se ve modificado constantemente debido a mejoras o a detalles susceptibles de modificación, así como contribuciones de la comunidad e incluso sugerencias que puedan surgir a partir de que el trabajo sea publicado, por lo que recomiendo consultar el repositorio donde éste se encuentra alojado, antes que utilizar el código publicado en el archivo digital de la tesis.

Mi objetivo como estudiante de posgrado en Tecnología Musical y a la vez como creador que se acerca a la investigación, es tener un compromiso no sólo con la obra final, sino también con el desarrollar productos tecnológicos a través del software (en este caso el libre), que resulten de la investigación académica y la experiencia práctica para poder establecer nuevos procesos que contribuyan a la mejora de los ya existentes. En este sentido el presente trabajo converge con la necesidad de reafirmar el desarrollo de herramientas musicales destinadas a ofrecer alternativas al software propietario que ha tenido gran impacto en el usuario y ha marcado estándares dentro del arte digital. También y por supuesto de forma importante, una motivación para realizar el trabajo es la necesidad natural de crear una herramienta que se adecuara a mis requerimientos al momento de hacer *laptop performance* mediante SuperCollider.

Capítulo I

Música por computadora

"La música está en todas partes, pero nunca ocupa el mismo lugar."
Jean Molino, *El hecho musical y la semiología de la música*

El artista es un ente social que se encuentra sujeto a un contexto espacio-temporal. La actualidad tecnológica forma parte de ese entorno, proporcionando herramientas que le facilitan la creación de una obra. Esta situación ha influido en la práctica artística durante las diferentes etapas de la humanidad, haciendo que la frontera entre el arte y la tecnología haya cambiado su línea divisoria a lo largo de la historia.

Para los antiguos griegos no existía una frontera distinguible entre una capacidad técnica y una artística, el vocablo *tekné*[1] servía para englobar ambas prácticas. Durante muchos años la humanidad lo entendió así. Berenguer (2002) incluso menciona: "el divorcio entre artistas y científicos tuvo su inicio con Newton y su modelo mecanicista del universo, y se consolidó a continuación con las consecuencias de su método, singularmente durante la Revolución Industrial" A medida que la sociedad acepta el paradigma Newtoniano y se toma a la razón y al método científico como bases para llegar al conocimiento, la comunidad artística empieza a poner distancia del quehacer de la ciencia.

> Así, mientras el científico juega con la realidad y la lógica, al artista le concierne la imaginación y la emoción. El arte investiga el mundo subjetivo; la ciencia, por su parte, persigue el mundo objetivo y el método racional. Como consecuencia de esta escisión, el mundo del arte acabó adoptando el romanticismo como ideología principal, y el artista se convirtió en un personaje marginal, un comentador y un crítico. (Berenguer, 2002).

Charly Percy Snow menciona en su ensayo *Las dos culturas* (1964, citado en Berenguer, 2002) la existencia de una división clara entre arte y ciencia, en esa frontera el científico juega con la realidad y la lógica, mientras que al artista le concierne la imaginación y la emoción; se considera, así, a la ciencia como objetiva y al arte subjetivo.

[1] τέχνη - "arte" y "ciencia" . (Orozco, 2007, p. 122).

En oposición a esta perspectiva hay posturas que predican una división mucho menos clara, un ejemplo fueron los movimientos vanguardistas surgidos a principios del siglo XX, los llamados “ismos”, en particular el futurismo, que planteaba la integración de la máquina como una herramienta artística. A pesar de ser un movimiento que no fue reconocido en su momento tuvo un fuerte impacto posteriormente, pues la idea de utilizar y explotar la tecnología surgida de la revolución industrial, en lugar de oponerse a ella, es retomada a lo largo del siglo XX por diversos artistas.

Dada su naturaleza novedosa, los avances tecnológicos estimulan la creatividad humana, de este modo, el uso de la tecnología en la práctica artística es un tema que por sí solo merecería ser tratado en extenso durante una tesis de grado, dado lo cual, nos parece relevante abordar el uso de la tecnología, y específicamente la computadora, en la práctica musical.

Es importante aclarar que partimos de una postura que entiende que la actividad artística no existe condicionada a estos procesos tecnológicos, pues aunque el medio técnico puede ser parte de, o ser la totalidad del mensaje que se busque comunicar, éste sería inútil si no se potencian estos progresos creando contenidos que aprovechen los nuevos medios. Es evidente que los avances logrados hasta ahora no son el fin del camino; hay nuevas tecnologías, por ejemplo, que ofrecen diferentes alternativas para el control del acto sonoro, como sería el caso de la gestualidad y la interacción que parecen opciones con mucho futuro. Sin embargo, para fines de esta investigación la computadora, y en particular el software, serán suficientes.

El uso de la computadora en la música ha sido constantemente discutido: es difícil establecer hasta qué punto la tecnología condiciona la estética y la técnica musical. Las posturas a este respecto pueden ser extremas e ir desde considerar al instrumento como parte sustancial del valor de la obra, prestando una atención especial al proceso de realización, o volcarse al otro lado desde la perspectiva en la que puede considerarse al ordenador como un artefacto que permite hacer de forma diferente lo mismo que se podría hacer con otros instrumentos.

> Según esta perspectiva, la computadora no produce música, como tampoco las plumas de ganso con las cuales Mozart escribía sus partituras han producido la música de Mozart... La pregunta es entonces saber si se puede definir una implicación estética del uso de la

> computadora —sin imputar a esta herramienta una importancia abusiva, y sin anular, tampoco, la especificidad de su papel, que es, sin embargo, diferente al de la pluma de ganso o del lápiz- esta pregunta es del mismo tipo, parece, que la que se hacía Walter Benjamin a propósito de otra innovación técnica: la de la reproductiblidad de la obra de arte. (Galard, 2000, p. 263).

Menciona Walter Benjamin (1973) que los cambios en la forma en la que se producen y reproducen los objetos artísticos están ligados a procesos (o autores) múltiples y no tanto a cuestiones individuales.

Partimos del supuesto que un adelanto técnico, como la computadora, no condiciona el proceso creativo. Si bien puede provocar modificaciones en el acto creador, ya sea en el tiempo de trabajo, el producto material, las formas de circulación, e incluso la valoración del objeto artístico; finalmente la computadora es solo un instrumento para el creador, que sigue siendo el responsable consciente del resultado. A final de cuentas, el contenido de la obra es lo que hará de ésta, un objeto perdurable o no. Es cierto también que el uso de las nuevas tecnologías se enfrenta a obstáculos qué resolver; por ejemplo, la obsolescencia tecnológica, que actúa como una restricción que reduce la vida de las obras que están atadas a sistemas específicos de una época; además de que en cada ocasión que surgen nuevas herramientas se vuelve necesario un período de adaptación para que se desarrollen especialistas capaces de crear resultados novedosos e interesantes que puedan desmarcarse de las tradiciones existentes, en este caso musicales. No suponemos que haya alguna especie de obligación de deslindarse de la tradición, pero sí queda un dejo de responsabilidad de los nuevos creadores, en su capacidad de usar medios que en el pasado no estaban disponibles.

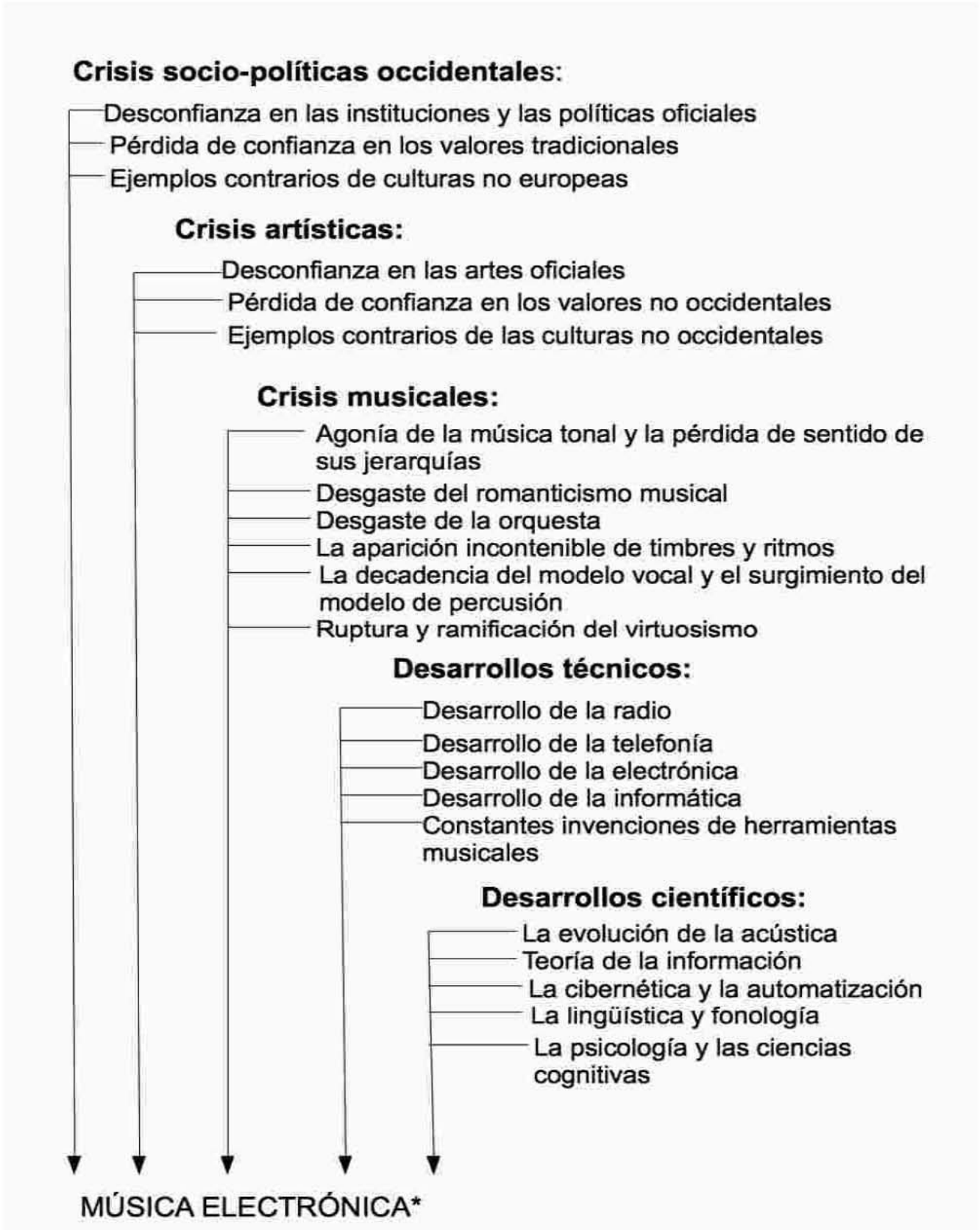
1.1 La Tecnología en la práctica musical a partir del siglo XX

En el caso de la música, los instrumentos electrónicos son una clara muestra del interés del ser humano por buscar nuevos sonidos y el control sobre distintas propiedades de los mismos: altura, timbre, etcétera. Ya desde 1876 Elisha Gray trabajaba en el desarrollo del electroarmonio, un piano capaz de mandar sonidos a través de la línea telefónica.

Más allá de la polémica alrededor de ser el primer instrumento electromagnético (algunos autores consideran al instrumento Theremin como el pionero) en este período de principios de siglo XX empieza a germinar en distintos laboratorios el interés por la creación

de instrumentos musicales que utilicen los avances en el campo electrónico. Si bien los adelantos técnicos dan un giro importante a las nuevas formas artísticas, resulta evidente que la música es uno de los campos, que desde la segunda mitad del siglo XX, ha sufrido transformaciones más radicales en la manera que el público general se acerca y relaciona a sus producciones artísticas. Son los desarrollos tecnológicos, medios de comunicación masiva e incluso la globalización misma, factores determinantes que nos permite afirmar que "la música ha alcanzado aspectos y dimensiones inéditas en épocas pasadas". (Ruesga, 2004, p. 5).

La siguiente imagen extraída de Bages (2011) muestra lo que Martin Laliberté observa como condiciones para la existencia de nuevas tecnologías musicales.


Crisis socio-políticas occidentales:
Desconfianza en las instituciones y las políticas oficiales
Pérdida de confianza en los valores tradicionales
Ejemplos contrarios de culturas no europeas
Crisis artísticas:
Desconfianza en las artes oficiales
Pérdida de confianza en los valores no occidentales
Ejemplos contrarios de las culturas no occidentales
Crisis musicales:
Agonía de la música tonal y la pérdida de sentido de sus jerarquías
Desgaste del romanticismo musical
Desgaste de la orquesta
La aparición incontenible de timbres y ritmos
La decadencia del modelo vocal y el surgimiento del modelo de percusión
Ruptura y ramificación del virtuosismo
Desarrollos técnicos:
Desarrollo de la radio
Desarrollo de la telefonía
Desarrollo de la electrónica
Desarrollo de la informática
Constantes invenciones de herramientas musicales
Desarrollos científicos:
La evolución de la acústica
Teoría de la información
La cibernética y la automatización
La lingüística y fonología
La psicología y las ciencias cognitivas
MÚSICA ELECTRÓNICA*

La composición musical contemporánea ha experimentado nuevos caminos para su construcción, el entorno de guerras mundiales y revoluciones durante el siglo pasado terminó por desechar el romanticismo como estilo predominante en el campo del arte; la sensación de progreso que se heredó de la revolución industrial fue borrada por el horror que causaron las guerras mundiales, y el mundo de la posguerra no experimentaba una sensación de avanzar en una misma línea.

La armonía tonal que era usada abre paso, sin desaparecer, a nuevas perspectivas a nivel composicional; aparece una fuerte tendencia a la composición atonal; destaca la obra de compositores como Debussy y Schöenberg, y algunas tendencias como el microtonalismo, politonalismo, dodecafonismo y posteriormente el serialismo, que además de modificar la altura de los sonidos juega con elementos como la duración, la intensidad o el timbre; aparecen, también, nuevos instrumentos, como el sintetizador y que dio pie al concepto de síntesis sonora. De este modo, inicialmente surge la síntesis aditiva derivada de la investigación de un matemático francés, Charles Fourier, pero posteriormente se irán incorporando otros tipos de síntesis, como lo es la FM o de frecuencia modulada desarrollada por John M. Chowning. (1973)

Estos nuevos conocimientos junto con otros ejercicios realizados en diversos laboratorios y centros de investigación contribuyeron al desarrollo de lo que posteriormente se conocería como música electrónica, entendida como aquella música que utiliza computadoras, sintetizadores o cualquier otro instrumento electrónico en su proceso creativo o interpretativo.

Diversos artistas empatan las revoluciones en las estructuras de la composición con el desarrollo de instrumentos y avances tecnológicos. Con la invención de la máquina de cinta, por ejemplo, se da una completa revolución que permite nuevas estructuras de composición como sucede en los trabajos de Pierre Schaeffer y Edgar Varése; así, provoca que la música electrónica sufra una evolución en los laboratorios y en las esferas académicas que se hace manifiesto en el correr de los años a nivel popular también.

La música electrónica junto a la música concreta y algunas otras vertientes, son amparadas por una sombra tecnológica que permite la experimentación sonora al punto de que la relación entre ciencia y arte se estreche tanto en laboratorios como en salas públicas.

Este proceso de liberación artística no fue exclusivo de la música, tenemos el ejemplo

de la literatura con el *beatnik,* la poesía no formal minimalista, entre otras. Estos cambios nos muestran que el proceso es resultado de una búsqueda que la sociedad experimenta: “La aparición de las nuevas tecnologías, y concretamente de las nuevas tecnologías musicales, nos revelan la confluencia de factores científicos, tecnológicos, sociales, filosóficos, musicales, etc. de nuestra época.” (Bagés, 2011, p. 3)

Cuestiones como la evolución de los soportes de grabación y los alcances de la distribución musical son elementos que también evolucionaron amparados en las nuevas tecnologías e influyeron en la práctica artística.

Desde mediados del siglo pasado aparece un instrumento tecnológico que se vuelve un protagonista esencial en el conjunto de fenómenos que analizaremos en el presente trabajo: la computadora.

De manera similar a lo que sucedió con los artistas de principios del siglo XX que encontraron en la maquinaria derivada de la revolución industrial nuevos instrumentos sonoros, la computadora encontró un acogimiento perfecto entre los artistas que buscaron nuevas formas de hacer música durante la posguerra. Con este desarrollo empezaron a existir una serie de actividades creativas facilitadas por la aparición de esta tecnología, que van desde el computo de datos para su uso en la composición de partituras, hasta la síntesis de sonido dentro de la propia computadora. En este sentido la fortaleza de la computadora es la capacidad de procesar datos y aplicarlos a todo tipo de resultado, más allá de la habilidad para simular un instrumento musical existente.

1.2 Uso de la computadora en la música: breve cronología de la música por computadora

A modo de hacer más fluido el recorrido histórico, a continuación presentamos una breve cronología[2] de la música por computadora, donde mencionaremos pasajes históricos, anexando una breve descripción para los que se consideran los más relevantes, y meramente enumerando el dato para aquellos que no son tan fundamentales.

[2] La cronología se ha realizado tomando referencias de diversas fuentes: Wang (2008), Jórda (2008), Manning (2004) y Doornbusch (2012).

1843

- ≅ Ada Lovelace una condesa británica especialista en matemáticas, escribió un artículo acerca de la Máquina Analítica de Charles Babbage (considerado el primer diseño de un computador moderno), en este artículo la condesa Ada escribió a manera de premonición las primeras ideas sobre la música por computadora.

> Suponiendo, por ejemplo, que las relaciones fundamentales de los sonidos...en la ciencia de la armonía y de la composición musical fueran susceptibles de tal expresión y adaptación, la Máquina podría componer piezas de música de cualquier grado de dificultad y extensión. (Berron, 2007, p. 10)

1887

- ≅ Emile Berliner patenta en Estados Unidos el gramófono lo que posteriormente da origen a los primeros discos de vinilo.

1907

- ≅ Ferruccio Busoni en su texto *Esbozo de una estética musical* deja constancia del agotamiento de los instrumentos tradicionales y de la necesidad de incorporar nuevos timbres, poniendo todas sus esperanzas en los instrumentos eléctricos que entonces ya empezaban a aparecer.

1910

- ≅ El compositor italiano Balilla Pratella, autor de la ópera “El aviador Dro”, publica el “Manifiesto de La Música Futurista.

1928

- ≅ Harry Nyquist desarrolla la teoría de muestreo de la señal analógica.

1937

- ≅ John Cage presenta en un congreso en los Estados Unidos su manifiesto “The future of music: credo” en el que habla del futuro de la música electrónica.

≅ Carlos Chávez publica *Hacia una nueva música* (publicado en inglés como *Toward a new music*) uno de los primeros libros en hablar de música electrónica.

1946

≅ Fue presentada al público la ENIAC que es un acrónimo de Electronic Numerical Integrator And Computer (Computador e Integrador Numérico Electrónico), construida en la Universidad de Pensilvania y considerada la primer computadora de propósito general.

1947

≅ Es inventado en los laboratorios Bell el transistor,un dispositivo electrónico semiconductor que cumple funciones de amplificador, oscilador, conmutador o rectificador, y que debido a sus múltiples aplicaciones revoluciona el mundo de la electrónica.

1948

≅ Se construye en la Universidad de Manchester Mark 1 (Baby) - la primer computadora con un programa almacenado - se comercializa como la Ferranti Mark I.

≅ Pierre Schaeffer crea un laboratorio de música concreta en la Oficina de Radiodifusión y Televisión Francesa (ORTF).

1949

≅ Se lanza UNIVAC I (UNIVersal Automatic Computer I) la primera computadora comercial en los Estados Unidos.

1951

≅ CSIRAC (Council for Scientific and Industrial Research Automatic Computer) construida en Australia se convierte en la primer computadora en reproducir música públicamente.

≅ La BBC realiza la primer grabación conocida de una computadora ejecutando música, grabando a la primera computadora de uso comercial en el reino unido la Ferranti Mark I.

1952

- ILLIAC I la primera computadora para uso educativo se construye en la Universidad de Illinois.
- Pierre Schaeffer publica e texto *À la recherche d'une musique concrète* (En busca de la música concreta).

1953

- Se comercializa la IBM 701, la primera computadora comercial de la empresa norteamericana.

1954

- La empresa IBM introduce en su modelo 704 su primer sistema operativo y completa la especificación para el primer lenguaje de programación de alto nivel, FORTRAN (Formula translation).

1955

- David Caplin y Dietrich Prinz programan la computadora Ferranti Mark I para empezar a investigar la generación de estructuras musicales, a partir de programar la interpretación de la obra de Mozart *Musikalisches Würfelspiel* (un método para generar valses a través del azar).
- Lejaren Hiller y Leonard Isaacson empiezan a trabajar en la Illiac Suite, el primer experimento de composición por computadora.

1956

- Martin L. Klein y Douglas Bolitho programaron la computadora Datatron para componer canciones populares, la canción Push Button Bertha con letra de Jack Owen es atribuida a esta computadora.

1957

- ≅ IBM presenta el primer compilador para FORTRAN.
- ≅ Lejaren Hiller y Leonard Isaacson completan Illiac Suite, primera pieza compuesta por un ordenador.
- ≅ Max Mathews escribe MUSIC I, programa de computadora para síntesis de sonido, usando formas de onda predeterminadas y digitalmente sintetizadas y un convertidor de digital a sonido, el primer uso musical de un Digital Audio Converter.

1958

- ≅ John McCarthy desarrolla el lenguaje de programación LISP.
- ≅ Se publica MUSIC II, la segunda versión del programa creado por Max Mathews.
- ≅ Francisco Kröpfl funda el primer centro de música electrónica en América Latina, el estudio de fonología musical en la Universidad de Buenos Aires.

1960

- ≅ Se publica MUSIC III y se convierte en el primer lenguaje modular para síntesis de sonido.

1962

- ≅ James Tenney compone *Four Stochastic Studies* (para computadora y cinta).
- ≅ Iannis Xenakis completa la serie ST (para cuarteto de cuerdas, ensamble y orquesta) usando la técnica de composición estocástica por computadora.
- ≅ Una de las primeras demostraciones a gran escala de música por computadora fue una emisión nacional de la NBC para el programa Monitor.

1964

- ≅ Se crea el código ASCII.
- ≅ El lenguaje de programación BASIC (Beginners All-purpose Symbolic Instruction Code), es desarrollado en Dartmouth College.
- ≅ John Chowning y David Poole empiezan a trabajar con Music IV en la Universidad de

Stanford en una IBM 7090.

1965

- La compañía Moog pone a la venta su primer sintetizador análogo de tipo modular.

1966

- Les Paul inventa la grabación multi-pistas.
- Pierre Schaeffer publica *Traité des objets musicaux*.
- Jean Claude Risset realiza investigaciones en los laboratorios Bell (EUA) acerca de la evolución de tonos de trompeta a partir de análisis rápidos de Fourier (FFT).
- Iannis Xenakis funda en París Francia el EMAMU (Equipe de mathématique et automatique musicales).

1967

- John Chowning descubre la frecuencia modulada (síntesis FM) cuando experimentaba con efectos extremos de vibrato.

1968

- Se lanza MUSIC V, escrito en FORTRAN.
- Jean-Claude Risset crea en los laboratorios Bell un catálogo de sonidos sintetizados por computadora utilizando MUSIC V.

1969

- Se transmite el primer mensaje a través de ARPANET y se establece el primer enlace entre las universidades de Stanford y UCLA, antecedentes de Internet.
- Ken Thompson y Dennis Ritchie desarrollan Unix en los laboratorios Bell.
- Se lanza el primer modelo de la familia de computadoras DEC PDP.
- John Cage y Lejaren Hiller colaboran en la composición HPSCHD aplicando principios de azar por medio del I Ching y un programa informático.
- Max Mathews y Moore desarrollaron GROOVE, un sistema de control digital a tiempo real para síntesis analógica.

1971

- ≅ Intel fabricó el primer microprocesador, identificado como 4004.

1972

- ≅ Es desarrollado el lenguaje de programación “C”.

1973

- ≅ El estándar Ethernet y la estación de trabajo “Alto”, uno de los primeros ordenadores personales de la historia que contaba con un mouse, son desarrollados por Xerox en PARC(Palo Alto Research Center).
- ≅ Primera conexión internacional de ARPANET (con Europa).
- ≅ Parte de Unix es reescrito en el lenguaje C.

1974

- ≅ Se lleva a cabo en la Universidad de Michigan State la primer Conferencia Internacional de Música por Computadora (ICMC por sus siglas en inglés).
- ≅ Curtis Roads desarrolla el primer sistema de síntesis granular por computadora en la Universidad de UCSD en San Diego California, usando una computadora Borroughs B6700 y el programa Music V.

1975

- ≅ John Chowning, James A. Moorer, John Grey y Loren Rush fundan en Stanford el CCRMA (Center for Computer Research in Music and Acoustics).
- ≅ Jean-François Allouis inicia en el GRM (Groupe de recherches musicales) de París, el proyecto Syter, sistema en tiempo real para audio digital (acrónimo de SYnthèse en TEmps Réel).

1976

- ≅ Laurie Spiegel crea VAMPIRE un sistema para video y composición sonora en tiempo real.
- ≅ Paul Berg desarrolla PILE un lenguaje para síntesis sonora en tiempo real en un ordenador PDP-15 en el Instituto de Sonología en Holanda.
- ≅ Steve Wozniak y Steve Jobs fundan la compañía Apple y sale a la venta el ordenador Apple I.

1977

- ≅ Pierre Boulez crea el Instituto de investigación y coordinación de música y acústica (IRCAM) en París, Francia.
- ≅ Es publicado por primera vez *The Computer Music Journal,* revista científica sobre la música por computadora.

1978

- ≅ MOUSE, software para síntesis y composición en computadoras es desarrollado por Peter Grogono.
- ≅ William Buxton ejecuta el software SSSP (Structured Sound Synthesis Project) en una computadora PDP- 11 controlando 16 osciladores digitales.
- ≅ Se desactiva el sistema GROOVE debido a que deja de operar la computadora Honeywell.
- ≅ James A. Moorer crea la primera aplicación de un Vocoder de fase en computadora.

1979

- ≅ Nace la Asociación internacional de música por computadora (ICMA por sus siglas en inglés), es fundada por Thom Blum, Curtis Roads y John Strawn.
- ≅ F. Richard Moore funda el Computer Audio Research Lab en la Universidad de California en San Diego y desarrolla el software CMUSIC.

1980

- ≅ Se presenta el disco compacto CD.

- ≅ Se crea en el IRCAM un sistema de espacialización de sonidos por computadora (4X) con el cual se estrena la obra “Repons” de Pierre Boulez.

1981

- ≅ Sony lanza el disco flexible o disquette de 3.5 pulgadas.
- ≅ Se presenta la IBM PC, usando un CPU Intel 8088 a 4.77MHz (0.33 MIPS).

1982

- ≅ Se lanza el ordenador Commodore 64 que incluía un chip llamado SID que permitía generar sonidos.
- ≅ Philips

y Sony comienzan a comercializar el CD. Primer sistema de grabación óptica digital

1983

- ≅ ARPANET se convierte completamente al protocolo TCP/IP dando paso al Internet moderno.
- ≅ ARPANET se divide en MILNET (para comunicaciones militares) y ARPANET (para aplicaciones civiles).
- ≅ Se publica la norma MIDI 1.0 (Musical Instruments Digital Interface).
- ≅ Se lanza comercialmente el primer sintetizador de uso popular que utiliza síntesis FM, el Yamaha DX7.
- ≅ El norteamericano Charles Dodge compone “Cascando”, una de las primeras composiciones por computadora que usan síntesis de voz.

1984

- ≅ SYTER 3 la versión completamente funcional del proyecto es publicada convirtiéndose en uno de los primeros sistemas de música por computadora para la transformación de sonidos en tiempo real en el INA-GRM en París.
- ≅ Sale a la venta la computadora Macintosh, con un procesador Motorola 68000, 512 kb de RAM y 64 Kb de ROM.

1985

- ≅ Microsoft lanza Windows 1.0.
- ≅ MacMIX es creado en el IRCAM por Adrian Freed para editar gráficamente los sonidos de un Macintosh conectado a una minicomputadora VAX 11/780.
- ≅ Richard Moore desarrolla CMUSIC, una versión expandida de MUSIC V en el CARL (Computer AudioResearch Laboratory), en la Universidad de California en San Diego.
- ≅ Charles Dodge y Thomas Jerse publican el libro *Computer Music*.
- ≅ Laurie Spiegel desarrolla Music Mouse un software de composición algorítmica para computadoras Macintosh, Amiga y Atari.

1986

- ≅ Barry Vercoe del MIT crea una versión del software MUSIC llamada Csound que puede ser compilada por cualquier computadora que soporte el lenguaje de programación C.
- ≅ IBM y MIPS Technologies presentan el primer Computador con Conjunto de Instrucciones Reducidas (RISC por sus siglas en inglés).
- ≅ Clarence Barlow crea Autobusk, software para composición algorítmica en el ordenador

Atari ST.

- ≅ Roger Dannenberg de la Universidad Carnegie Mellon presenta CMU MIDI Toolkit una librería de programación para escribir software MIDI interactivo en lenguaje C.
- ≅ Tod Machover establece el grupo de investigación de Hyperinstrumentos en el MIT.
- ≅ Barry Truax desarrolla la primer implementación en tiempo real de síntesis granular con un procesador de sonido DMX-1000.

1988

- ≅ Miller Puckette desarrolla Patcher (después llamado Max) en el IRCAM para controlar el sintetizador 4X.
- ≅ Sale a la venta la computadora NEXT, con una velocidad de 25 Mhz y 64 Mb de RAM.

1989

- ≅ La compañía Digidesign en Estados Unidos comercializa el primer programa de audio digital estéreo (Sound Tools) para Macintosh.
- ≅ Se presenta CUBASE 1.0 secuenciador MIDI para Atari. La primer versión introduce el concepto de *arrange pages,* una lista vertical que contiene las pistas y en la parte horizontal la línea de tiempo, diseño que rápidamente se convirtió en la interfaz estándar para el desarrollo de secuenciadores comerciales.
- ≅ El sistema World Wide Web es creado por el inglés Tim Berners-Lee con la ayuda del belga Robert Cailliau mientras trabajaban en el CERN en Ginebra, Suiza aunque es publicada hasta 1992.

1990

- ≅ Se presenta la tarjeta de sonido Digidesign Audiomedia I, para DSP y audio I/O en computadoras Apple Macintosh.
- ≅ La compañía Opcode comercializa el programa MAX creado por Miller Puckett en el IRCAM.

1991

- ≅ Se publicó el primer documento formal con la descripción de HTML para ser consultado en línea bajo el nombre HTML Tags (Etiquetas HTML).
- ≅ Iannis Xenakis completa GENDYN un software dinámico para síntesis estocástica.
- ≅ Se lanza la primer versión del software Pro-Tools.
- ≅ Tom Erbe lanza SoundHack DSP, software para manipulación de sonido.
- ≅ Se presenta Symbolic Composer, software para composición algorítmica.
- ≅ Se crea en el IRCAM el programa SVP, antecedente de AudioSculpt.

1991

- ≅ Se lanza la tarjeta de audio Sound Blaster Pro que permitía grabar a una frecuencia de muestreo a 22 Khz y reproducir sonido digitalizado hasta a 44.1 Khz, además de incluir un mezclador para controlar volúmenes.

1992

- ≅ Windows 3.1 se pone a la venta.
- ≅ Apple introduce la tecnología Quicktime media.
- ≅ IBM lanza su primer computadora laptop llamada ThinkPad.
- ≅ SensorLab (sensor para interfaces MIDI) es lanzado por STEIM.

1993

- ≅ Emagic Notator Logic un secuenciador MIDI y DAW (Digital Audio Workstation) es lanzado para Macintosh.
- ≅ Se lanza al mercado el primer procesador Intel Pentium.

1994

- ≅ Se lanza Linux 1.0
- ≅ La compañía SEYYO lanza el software Koan para crear música generativa.
- ≅ Se introduce al mercado la PowerMac de Apple que marca grandes avances en cuestión de hardware al usar procesadores PowerPc 601, desarrollados en conjunto por Apple, IBM y Motorola, la mejora en rendimiento se debía principalmente al uso de arquitectura RISC (Reduced Instruction Set Computing).
- ≅ IRCAM crea AudioSculpt, un programa que permite al usuario manipular sonido basado en la tecnología Super Phase Vocoder (SVP) usada también por el IRCAM para recrear una voz *Castratti* para la película Farinelli.

1995

- ≅ ARPANET es renombrado como Internet.
- ≅ Aparece Windows 95, sistema operativo con interfaz gráfica (englobando MS-DOS y Windows 3.XX), es tal su éxito que llega a acaparar el 70% del mercado de computadoras personales.
- ≅ El software LiSa (live sampling) es lanzado por la compañía STEIM.
- ≅ Se crea Cynthia interfaz gráfica para el programa Csound.

- ≅ Karlheinz Brandenburg utiliza por primera vez la extensión .mp3 para archivos de sonido, un formato de compresión digital que usa un algoritmo con pérdida para conseguir un menor tamaño. Alcanzó gran popularidad gracias a Internet y a que permitió el intercambio de ficheros musicales.

1996

- ≅ Native Instruments lanza el software para síntesis Generator 0.96 (primera versión de Reaktor).
- ≅ Steinberg incorpora la tecnología VST (Virtual Studio Technology) dentro de Cubase, una interfaz que permite conectar sintetizadores y plugins a editores de audio y sistemas de grabación.
- ≅ James McCartney diseña SuperCollider un lenguaje de programación orientada a objetos que se utiliza como herramienta para el procesamiento de audio en tiempo real y composición algorítmica.
- ≅ Perry Cook presenta STK (Synthesis Toolkit) una librería para síntesis de sonido escrita en C++.
- ≅ Miller Puckete (creador original de Max/MSP) presenta Pure Data (o Pd) en la ICMC, un lenguaje de programación gráfico para creación de música interactiva y obras multimedia que sigue el precepto del software libre.

1997

- ≅ Se agregan las extensiones de audio a Max y se convierte en Max/Msp.
- ≅ Se presenta el protocolo OSC (Open Sound Control).
- ≅ ASIO (Audio Stream Input/Output) protocolo de ordenador para audio digital y VST se lanzan como estándares abiertos, lo que permite a desarrolladores externos crear plugins.

1998

- ≅ Se lanza AudioMulch software modular para crear música y procesar sonido.
- ≅ Andrew Sorensen y Andrew Brown presentan jMusic una librería para composición

musical escrita en Java.

- ≅ Se crea OpenMusic (OM) un ambiente visual de programación orientado a objetos para composición musical basado en Lisp.

1999

- ≅ El estandard VST de Steinberg incorpora por primera vez instrumentos virtuales, lo que a la larga permitirá virtualizaciones de instrumentos reales como el sintetizador Moog por ejemplo.
- ≅ IRCAM y CNMAT desarrollan SDIF (Sound Description Interchange Format).
- ≅ Shawn Fanning y Sean Parker publican la primera versión de Napster una tecnología que permitía el intercambio de archivos en formato MP3 entre sus usuarios.

2000

- ≅ Propellerheads lanza la primera versión del software musical Reason.
- ≅ Se lanza Loris, software *open source* para modelado de sonido.

2001

- ≅ Se crea el formato AAC (Advanced Audio Coding).
- ≅ Se pone a la venta el Ipod de Apple.
- ≅ Native Instruments publica la primera versión del software Ableton Live.
- ≅ Se lanza JMSL (Java Music Specification Language) lenguaje de programación para creación musical basado en HMSL.
- ≅ Se funda la organización Creative Commons.
- ≅ Se presenta la primera versión de Processing.

2002

- ≅ Se lanza la versión 3 de SuperCollider que incorpora un cliente y un servidor, se convierte en *free* software.
- ≅ Apple compra Emagic y toma el control de la aplicación para audio Logic.
- ≅ Software Junxion (Sensor de MIDI) publicado por STEIM.

2003

- ≅ Se publica ChucK lenguaje de programación para síntesis sonora y composición musical en tiempo real, creado en la Universidad de Princeton.
- ≅ Inicia el proyecto Reactable en la Universidad Pompeu Fabra de Barcelona, para desarrollar un instrumento electrónico colaborativo y modular, utilizando una mesa como interfaz y desarrollada en Pure Data, será publicado dos años después.
- ≅ Comienza a operar la Itunes Music Store, para venta de archivos musicales a través de Internet incluyendo MP3.
- ≅ Se publica la extensión para facilitar la modificación de código en tiempo real, JITLIB (Just in Time Library) en SuperCollider.
- ≅ Se empieza a comercializar el disco Blu-ray.

2005

- ≅ Se presenta Arduino, una placa electrónica con microntrolador que se distribuye como plataforma de hardware libre.
- ≅ Andrew Sorensen lanza el software *Impromptu*, para *live coding*.
- ≅ Se funda la Princeton Laptop Orchestra.
- ≅ Se lanza Fluxus un ambiente de programación en tiempo real para gráficos 3d, sonido y juegos.

2007

- ≅ Se pone a la venta el iPhone de Apple un smartphone con sistema operativo.
- ≅ Apple lanza Logic 8 con capacidad de hasta 255 pistas de audio.

2008

- ≅ Aparece RjDj, que permite la ejecución de archivos de Pure data en un iPhone.
- ≅ Se lanza la aplicación TouchOSC, que permite implementar sistemas gráficos personalizados para el control táctil entre aplicaciones que envían y reciben mensajes OSC.

2009

- ≅ Se presenta Tenori-on un instrumento musical electrónico diseñado por Toshio Iwai y Yu Nishibori que consiste en una matriz de 16x16 interruptores LED, que funcionan como controlador musical permitiendo asignar a cada uno una nota o un sonido.
- ≅ Se presenta Eigenharp, instrumento musical y controlador que incluye teclado, boquilla para alientos y botones para percusión.

2010

- ≅ Se lanza el controlador Kinect para Xbox.
- ≅ Apple presenta su tableta iPad.

2011

- ≅ Se lanza Kinect compatible con PC.

2012

- ≅ Se presenta el sensor de movimiento para computadoras Leap Motion.

1.3 Cambio de paradigmas en la práctica musical producto de las nuevas tecnologías

La relación entre música y tecnología se ha hecho mucho más estrecha a partir del desarrollo de los medios de grabación y reproducción, si bien el ámbito de la fabricación de instrumentos siempre fortaleció este vínculo, la relación que ofrece hoy la computadora con el ámbito musical es más fuerte que nunca.

Hasta la década de los ochenta, los experimentos artísticos con ordenadores se reservaban a laboratorios y centros académicos que contaban con el soporte económico para tener una computadora que permitiera ese tipo de experimentos, como la especialización para saber manejar los instrumentos que se fueron construyendo durante las décadas de los sesenta y los setenta. Durante esos tiempos los equipos tenían que programarse en lenguajes complejos, y fue hasta mediados de los años ochenta, que en algunos países los equipos computacionales se volvieron accesibles en precio y tamaño,

permitiendo que se considere a las computadoras como una herramienta útil para hacer arte.

Fue durante la década de los noventa que se integran los ordenadores al uso en los hogares de forma relativamente común, debido a la creciente popularidad de la computadora personal y a la masificación de Internet como medio de comunicación público. Mucha de la gente que trabajó con las computadoras en esta década creció aprendiendo lenguajes de programación como Basic, Pascal o C, usando ordenadores como Sinclair, Commodore o Atari y su secuenciador MIDI Notator. Años después, software como Photoshop, Premiere, Illustrator, Cubase y ProTools experimentaron una explosión comercial y profesionales del diseño, música y cine comenzaron a darle un uso profesional a las computadoras. A nivel musical, las capacidades del software para la organización de muestras o la creación de ritmos complejos y exactos que pocos bateristas serían capaces de ejecutar, empezó a generar tendencias sobre todo en la música pop. Así, la música por computadora se agrega al entramado de la expresión musical, entendiendo esta estructura en sí misma como un sistema en su definición.

> Un sistema no es un agregado, ni una mera suma de elementos yuxtapuestos mecánicamente. Por el contrario, sus elementos constitutivos, lejos de tener un valor como entidades independientes, presentan un carácter puramente relacional, es decir, dependiente de las relaciones que los unen y oponen con los demás en el seno de un conjunto. Un conjunto asume las características de sistema, si es posible concebir, por la vía de un modelo, las relaciones que otorgan a los términos que unen un valor de posición. Por la cuál constituyen una totalidad articulada... Se deben, pues reconocer las relaciones complejas que, en su operación constituyen a los elementos – de entrada separables- como partes de una totalidad postulada previamente. (Jáuregui, 1987, p. 99).

Estas relaciones complejas son muy diversas y se pueden abordar desde distintas perspectivas y prioridades, ya que van desde el cambio en la relación compositor–instrumentista, la colaboración y la reconstrucción de obras, así como la distribución y creación de herramientas a partir de mecanismos que las nuevas tecnologías permiten, siendo esta última idea la que se presentará en los siguiente capítulos.

“Plantea Nicholas Cook que la música es un modo de crear significado, y no sólo de representarlo... entonces podemos verla como un medio de mover nuestra propia posición, construyendo y reconstruyendo nuestra propia identidad cultural en el curso del proceso.”

(Ruesga, 2005, p.12) Por lo tanto, al ser la práctica musical trastocada por la tecnología y los procesos que la rodean, es importante conocer al interior estos procesos, ya sea para criticarlos, adherirlos a nuestros modelos de creación o incluso categorizarlos. Vale tratar de evitar lo que Christopher Small mencionaba:

> Tenemos miedo de enfrentarnos con experiencias musicales nuevas, en las que el conocimiento y la condición de expertos no nos sirven para orientarnos, y solamente puede servirnos la vivencia subjetiva, sentida con tal honestidad, y por eso nos refugiamos en la seguridad del pasado, donde sabemos qué hemos de esperar, y donde lo que más importa es la condición de *connoisseur*. (Small, 1989, p. 15).

Así, adentrándonos en estas formas novedosas se podrá llegar a una mejor comprensión de las nuevas herramientas musicales; no permitirnos que algún tipo de estancamiento afecte la práctica, mediante la asimilación de procesos que permitan replanteamientos, rupturas; y también una revaloración de las construcciones históricas, generando nuevos modelos de creación musical y una sana dinámica que podría servir en otros ámbitos de la vida moderna, ofreciendo, incluso, rasgos que ayuden a mejorar la identidad cultural de la sociedad contemporánea.

1.4 La música por computadora en la actualidad

Si se piensa en las computadoras de uso común, las aplicaciones en el ámbito musical representan uno de los puntos más interesantes tanto por el grado de complejidad que llegan a alcanzar como por la accesibilidad; los programas más amables al usuario representan un campo de fácil aprendizaje dentro de la música por computadora. La habilidad de programar, junto con conocimientos musicales sólidos, ofrecen una ventaja para obtener resultados significativos a niveles más complejos en este campo. En el futuro cercano es muy factible que el usuario promedio construya sus propios sistemas, incluso a nivel de hardware, lo que vislumbra un futuro bastante promisorio para la música por computadora.

Se puede hablar de la computadora como un instrumento de ayuda para la creación musical en dos sentidos: por un lado como creador del sonido, partiendo del concepto de “síntesis sonora” que consiste en “obtener sonidos a partir de medios no acústicos,

variaciones de voltaje en el caso de la síntesis analógica o por medio de programas de computadora en el caso de la síntesis digital", (Pajares, 2011, p. 520) con la intención de simular sonidos, crear orquestaciones y presentaciones en directo. Por otro lado el ordenador también es un dispositivo que se convierte en un elemento en la cadena de composición, pudiendo servir a distintos fines dentro de esta cadena, funciona como secuenciador y como un dispositivo que compone automáticamente a partir de patrones establecidos por el usuario, realizando cálculos que pueden ser demasiado complejos para el ser humano. En este sentido los primeros secuenciadores por computadora fueron creados para reproducir sonido muestreando instrumentos análogos, paulatinamente con el aumento de la potencia en los procesadores ha aumentado también la capacidad de estos secuenciadores.

La música por computadora desde sus inicios utilizaba programas cuyo núcleo consistía en formas de generar música usando matemáticas complejas, algoritmos que son capaces de ejecutar en un instrumento secuencias imposibles de tocar para un ser humano, así como exploraciones de escalas y tonos que no eran comunes o se consideraban imprácticas por los excesivos cálculos y la rapidez necesaria, e incluso sistemas aleatorios.

Este núcleo se mantiene en la actualidad, se siguen empleando técnicas que tienen su origen en los años sesenta y setenta.

A estas técnicas sumamos hoy en día, el deseo de personalizar las herramientas, la flexibilidad del software permite que los artistas-programadores prácticamente escriban su propio software para la creación, tanto musical como multimedia en general, generando así herramientas personalizadas, las cuales por sí mismas, son también un producto artístico.

Esta producción por parte del usuario va desde software generativo como Auto-Illustrator de Adrian Ward[3] a extensiones y *plug-ins* para programas como Max/MSP, entre muchos otros ejemplos, algunas empresas han producido incluso software que permite al usuario crear sus propios instrumentos, a partir, por ejemplo, de programación gráfica como el caso de la empresa Native Instruments con su programa Reaktor, donde el usuario es capaz de diseñar sus propios instrumentos, secuenciadores, *samplers* y otras aplicaciones que pueden comunicarse con dispositivos externos como teclados, controladores o equipos con algún sensor que pueda traducir los datos obtenidos en información digital. Desde luego también se encuentra la alternativa de la producción de herramientas por medio de software

[3]Referencia: http://swai.signwave.co.uk/

libre usando programas como SuperCollider, lo que se ahondará en los capítulos siguientes.

Estas nuevas perspectivas remiten a lo que Kim Cascone (2000) menciona haciendo eco de Marshall McLuhan, "el medio ya no es el mensaje; se ha dado paso a un nuevo fenómeno, las herramientas en sí mismas se han convertido en el mensaje"[4], la computadora para muchas disciplinas entre ellas la música se ha convertido en una herramienta y un medio que requiere un estudio detallado sobre lo que puede traerle a la música.

Capítulo II

Software libre en la práctica artística

"There ought to be an artistic depot where the artist need only hand in his art-work in order to receive what he asks for."
Ludwig V. Beethoven en *Bethoveen's Letters*

2.1 Implicaciones sociales del libre intercambio cultural

Durante las primeras décadas del siglo XX se dio el florecimiento de los ismos: futurismo, dadaísmo, cubismo, entre otros, los cuales proponían una renovación del arte y un cuestionamiento de la función social del mismo. En esos momentos la composición del mundo en los distintos niveles, político, social y económico, permitió una dinámica de transmisión cultural por la cual la información de estos movimientos llegó con poco tiempo de diferencia a diversas ciudades, dicho proceso fue acelerando su velocidad en el transcurso del siglo lo que hizo que incluso ramas como la antropología pusieran interés en los procesos de transmisión cultural y las modificaciones que han sufrido los mismos.

Durante los últimos veinte años la expansión de los medios electrónicos ha permitido que la información se transmita a velocidades tan rápidas que prácticamente en tiempo real nos enteramos de los acontecimientos en el otro lado del mundo. Los soportes de información han dejado de ser análogos y se han vuelto digitales. Se han afectado las

[4] Original en inglés: "Marshall McLuhan's The medium is the message," has given way to a new phenomenon: "The tool is the message".

condiciones en las que se produce, distribuye y accede a la cultura, fruto de las nuevas tecnologías en conjunto con las prácticas sociales, nos enfrentamos a una condición de novedad en las formas de producción-transmisión cultural. Es importante como miembros de la sociedad moderna fomentar el libre intercambio de información, para a su vez permitir la diversidad cultural y una producción de contenidos libres que sean susceptibles de modificaciones, como una forma de preservar libertades básicas en las sociedades actuales.

En contraste a las condiciones sociales, las legislaciones y los tratados internacionales de comercio han rechazado propuestas cuya "idea era clara y evidente: eliminar el copyright de allí donde no está haciendo nada salvo bloquear el acceso y la difusión del conocimiento" (Lessig, 2005, p. 252) obedeciendo a intereses económicos que no necesariamente van en mejora del autor sino de los mecanismos e industrias que intentan monopolizar la distribución, por ejemplo, de bienes culturales. Esta cuestión es abordada a detalle en el trabajo de Lawrence Lessig (2005) "Por una cultura libre. Cómo los grandes grupos de comunicación utilizan la tecnología y la ley para clausurar la cultura y controlar la creatividad".

Una revisión al origen de los sistemas de patentes, demuestra que la intención original de los mismos era promover la autoría y generar mecanismos que permitieran exponer los detalles de las invenciones y así animar a otros inventores a interesarse en lo que se publicaba, es decir, se pensaba en el bienestar de la sociedad como conjunto y no sólo en los intereses individuales o de alguna empresa. El paso del tiempo muestra claramente que el uso de estos conceptos se ha inclinado por otro tipo de intereses, en Estados Unidos por ejemplo, la primer concesión de *copyright* establecía una vigencia de catorce años con opción de extenderlo otros catorce, periodo que al concluir liberaba la obra como parte del dominio público. Durante el siglo XX está vigencia fue aumentando gradualmente hasta que desde 1998 el período que requiere una obra para volverse de dominio público es de noventa y cinco años a partir de su publicación.

La existencia de oligopolios en los medios de comunicación masivos y en los medios de producción promueve la búsqueda de mecanismos prohibitivos que funcionen para proteger una obra o una herramienta, mientras que la existencia de canales públicos y comunitarios es un ejemplo de alternativas que podrían buscar tanto las instituciones como la población en general. Trasladado al ciberespacio se tiene el ejemplo de la creación colectiva

en Internet, donde diversos usuarios son productores y distribuidores de contenidos. Si en el siglo XX los medios de comunicación modificaron la velocidad de transmisión de los contenidos culturales, actualmente son necesarios mecanismos que también modifiquen la intención de privatizarlos, los contenidos distribuidos mediante la red no pueden estar sujetos a un sistema controlador, no debe haber dueños únicos de los modos de producción, sino una comunidad donde todos pueden llegar a contribuir en la creación del contenido. No se debe olvidar que el artículo 27 de la Declaración Universal de los Derechos Humanos dice: “Toda persona tiene derecho a tomar parte libremente en la vida cultural de la comunidad, a gozar de las artes y a participar en el progreso científico y en los beneficios que de él resulten.” La conformación actual del mundo y de las nuevas tecnologías, obliga a llevar a la práctica estos preceptos, la historia del ser humano ha demostrado, debido a la importancia que tiene el artista en la sociedad, que el arte es también un servicio público. No se puede olvidar, por ejemplo, que el músico tiene una función social como creador de arte:

> La música tiene pues una naturaleza auténtica determinada por los medios sociales, culturales y económicos, esta sujeta a los cambios individuales y esta bullendo en relaciones de significación que dependen de una enorme multitud de factores. Por ello su función primera no debe ser buscada en la estética abstracta, sino en la eficacia de su participación en una regulación social históricamente determinada”. (García, 2002, p. 158) .

El impulso tecnológico de nuestros días ha generado comunidades virtuales que comparten recursos y proyectos a través de la red, que ayudan al usuario a incorporarse a un nuevo modo de creación, sin importar si tiene una preparación previa o no, se comparten piezas de arte pero también tutoriales, videos y explicaciones sobre las formas de creación, lo que lleva a pensar en una democratización de la producción artística. Seguir este tipo de modelos puede llevar a que la circulación del conocimiento se transforme de forma similar a como se modificó la circulación de personas en la época de la primer revolución industrial o el flujo de información con los primeros medios de comunicación masivos.

> La posibilidad de acceso a nuevas tecnologías por una mayor cantidad de usuarios, repercute en mas producción, mas propuestas y deriva en una potencial exploración sonora que a su vez puede devenir en una fabulosa experiencia estética y cognoscitiva. Esta es una época en la que la tecnología digital es un bien doméstico, solo debemos definir, nada mas y nada menos, que queremos hacer con ella. Consecuentemente nos acercamos a producción de arte que tiene

> menos tintes de institucional y es cada vez más personalizada, son ideas que “progresivamente” han ido ganando espacio en la producción contemporánea a partir de la idea de “hazlo tu mismo. (Haro, 2006, p. 45).

Hoy, en buena parte gracias a las dinámicas demográficas, hay nuevas propuestas artísticas, demostrando que las condiciones culturales llevan una relación profunda con la dinámica social “El nuevo paradigma tecnológico es el fiel reflejo cultural de las sociedades que desarrollan la tecnología.” (Vercerlli, 2004, p. 4) He ahí la razón por la que la transmisión de contenidos culturales y herramientas artísticas va de la mano con estos nuevos paradigmas tecnológicos. No se debe olvidar que los avances tecnológicos y técnicos han fungido a lo largo de la historia como reflejo y motivo de diversos cambios en las relaciones del arte con los demás ámbitos de la sociedad, en la música, por ejemplo:

> El efecto de la notación musical, y su posterior registro, sobre la organización del universo estético sonoro, que no sólo modificó la composición, interpretación y reproducción musical, sino que además trastocó el propio estatuto de la música, al desplazarla desde su función ritual originaria hacia el producto mercantil que es hoy en día. (Alonso, 2009, p. 42).

Las culturas de la antigüedad, como las Mesoamericanas, transmitían sus cantos y rituales musicales por medio de la oralidad, a raíz del intercambio cultural que supuso el descubrimiento y conquista de América se aceleró el proceso de integración y conocimiento entre las diversas culturas musicales que existían en ambos lados del Atlántico, lo que permitió conectar diferentes tradiciones que dieron origen a géneros y ritmos muy diversos. Está mecánica de conocimiento de las tradiciones musicales a través del contacto directo perduró durante varios siglos, todavía a principios del siglo XIX la música tenía que presenciarse en vivo para ser escuchada. No fue sino hasta la invención del fonógrafo que se experimentó un cambio en la transmisión de los contenidos musicales, ya que al momento de poder almacenarse en un formato físico, la música llegó a una mayor cantidad de personas. Incluso poblaciones fuera de los centros urbanos tuvieron acceso a información que no hubieran tenido sin la música grabada; hoy, a través de Internet está sucediendo algo similar al acercar tecnologías, software, y desde luego música a sectores de la población que en otro contexto no tendrían acceso.

> Con el desarrollo tecnológico operado en los últimos cien años, la llegada de los medios de comunicación de masas y a la vez cierto grado de democratización de la cultura, facilitado por la aparición de una industria y un mercado masivos, la música ha alcanzado aspectos y dimensiones inéditas en épocas pasadas. Como consecuencia, se ha producido un espectacular aumento de sus "usuarios" y una transformación radical en la forma en que nos relacionamos con ella. (Ruesga, 2004, p.5).

Actualmente encontramos a productores, *dj's* y artistas sonoros que son capaces de compartir música bajo esquemas diferentes a los que la industria tradicional permite. La cultura musical transmitida a través de la Web 2.0 trasciende cuestiones del lugar de origen o los antecedentes, la música se adapta a regiones geográficas y tradiciones, forma parte de un contexto global que por igual encuentra espacios para germinar en una ciudad africana que en un suburbio de Europa.

La música, enmarcada en las nuevas dinámicas de transmisión, permite al receptor ser un sujeto activo.

> El receptor no es un sujeto pasivo que recibe un determinado mensaje. Por el contrario, el sujeto que "recibe", desempeña un papel activo en la manera de recrear e interpretar ese mensaje. En este caso, a partir de una pieza musical (la huella), el escucha verá afectadas sus pasiones y las proyectará en la obra percibida. (Camacho, 2001, p. 6).

El software y el contenido de la red proporcionan medios para que personas de diversos estratos o regiones del mundo puedan acercarse relativamente con la misma facilidad a la vanguardia de la producción musical. Esto nos lleva a la colaboración que permite también el desarrollo tecnológico para que la comunidad pueda crear sus propias herramientas.

Dentro de la dinámica capitalista, se entiende que los artistas de vanguardia son los que van moldeando al público, sin embargo, hoy en día ese público por medio de los nuevos mecanismos de transmisión se convierte rápidamente en el nuevo artista de vanguardia, esta idea es sólo una más de las que permiten exponer la importancia de analizar estos procesos.

El siglo XXI e Internet ofrecen un mundo donde la cultura, incluyendo la música, está siendo absorbida y reutilizada, han surgido términos como *User generated content* -contenido generado por el usuario- o *Fair Use* -uso justo- término utilizado por la legislación sobre derechos de autor en algunos países principalmente en Estados Unidos, que permite el uso

limitado de material protegido sin necesidad de tener permiso del dueño del Copyright, siempre y cuando se justifique la razón de utilizar este material. Un ejemplo de este caso, es para el uso académico o informativo. Esta responsabilidad social es abanderada por personas detrás de la creación de nuevas tecnologías digitales pero también es responsabilidad del artista y específicamente de la música como disciplina artística que corresponde a una sociedad.

> Las estructuras sociales como las musicales son productos de los procesos cognitivos particulares de una cultura, indicando una estrecha relación estructural entre las normas de la organización social y musical. Según este enfoque, el sistema musical se compone de una estructura sonora y una función sociocultural particular. (Nettl, 1984 y Blacking, 1973, citados en Adje, 2005, p.3).

Las posibilidades son infinitas y trascienden la práctica musical. Actualmente hay diversas muestras que dan una idea de las ventajas de estos modelos constantemente retomados en el ciberespacio, por ejemplo, algunas de las universidades más importantes a nivel mundial ofrecen sus clases magistrales de forma abierta a través de la transmisión en línea, además existen proyectos como Conexions.[5] Por otro lado, tenemos proyectos de investigaciones científicas que integran a usuarios de todas partes del mundo para generar préstamos masivos de CPU, la transparencia de información burocrática que abandera Wikileaks y un largo etcétera. Pero hay una práctica que es ya bastante difundida y que abandera este tipo de creación colectiva, promoviendo el libre acceso a la información en beneficio de la sociedad y de la libre transmisión cultural, el software libre, del que por supuesto, se ahondará en las siguientes páginas.

2.2 Un poco de historia sobre software libre

A finales de los 60 un grupo de empleados de los laboratorios Bell de AT&T, entre ellos, Ken Thompson, Dennis Ritchie y Douglas McIlroy desarrollaron un proyecto llamado UNICS (Uniplexed Information and Computing System), Unix como sería registrado oficialmente; un sistema operativo portable, multitarea y multiusuario. El mismo grupo de desarrolladores creó

[5]Connexions comparte junto con otros proyectos como MIT OpenCourseWare y la Public Library of Science (Biblioteca Pública de la Ciencia), en la idea de que el contenido educativo y escolar puede y debe ser compartido, reutilizado y recombinado, interconectado y continuamente enriquecido.

años después el lenguaje de programación C que sirvió para reescribir buena parte de Unix en otras plataformas, las licencias de Unix fueron otorgadas a universidades y empresas para que hubiera diversos desarrolladores trabajando en la mejora y estabilidad del mismo, lo que convirtió al proyecto en una mezcla de un modelo comercial y académico. Esto también dio pie a ciertas disputas de derechos intelectuales puesto que no todos los desarrollos tenían la misma finalidad y para los años ochenta había diversas versiones del sistema Unix, muchas de las cuales eran sistemas de código abierto. Unix fue un primer intento de apertura de código fuente en sistemas operativos; desde un enfoque comercial, no logró establecer unificaciones efectivas, pero ayudo mucho a que se originaran nuevos proyectos. Algunas de estas ideas nuevas se cuenta la del 27 de septiembre de 1983, cuando se anunció la fundación del proyecto GNU,[6] iniciado por Richard Stallman con la intención de crear un sistema operativo completamente libre, pues para él "Un sistema operativo no significa sólo un *kernel* (núcleo), que apenas permite ejecutar otros programas." (Stallman, 2004, p. 19) El proyecto incluyó la publicación de un manifiesto que se puede leer en su página oficial[7] y desembocaría un par de años después en la creación de la *Free Software Foundation*. Dentro del manifiesto publicado se da a conocer la licencia llamada *Copyleft* que establece como principio que el programa en cuestión puede ser utilizado, modificado y distribuido sin ninguna restricción pero a su vez todos los productos derivados conservan estas libertades. "De este modo, las libertades cruciales que definen el «software libre» quedan garantizadas para cualquiera que posea una copia; estas libertades se convierten en derechos inalienables." (Stallman, 2004, p. 24)

Los ochenta fueron una década en la que se vivió una fuerte expansión de la informática y que también mostró que la estandarización del software no necesariamente era lo ideal para el avance de la tecnología. En ese momento existían dos protocolos principales para la comunicación entre redes: OSI y TCP/IP, el primero buscaba estandarizar la comunicación con el objetivo de colaborar al desarrollo industrial, el segundo de código abierto y uso libre, tenía la meta de compartir recursos informáticos. Fue dentro del segundo protocolo que se instauró la *world wide web* lo que posteriormente facilitó también el alcance masivo de Internet. "Una planificación y regulación gubernamentales (frente a la actividad empresarial y la libertad individual), que quizá esté representado a la perfección en las

[6] GNU es un acrónimo recursivo que significa GNU No es Unix (GNU is Not Unix).

[7] http://www.gnu.org/gnu/manifesto.es.html

figuras emparejadas y opuestas de Friedrich Hayek y Maynard Keynes." (Kelty, 2008, p. 147)
Para los años noventa la proliferación de marcas en el hardware generó una saturación de programas y problemas de compatibilidad entre dispositivos, por lo que el software libre y abierto emergió como una posible solución a este tipo de problemáticas. Fue precisamente en 1991 cuando Linus Torvalds desarrolló un *kernel* -núcleo- compatible con el sistema Unix y lo llamó Linux. En el año de 1992, la combinación de Linux con el sistema GNU resultó en un sistema operativo libre llamado, GNU/Linux, uno de los proyectos más exitosos del software libre, cuya intención se puede resumir en las siguientes palabras:

> Más que el simple ahorro del precio de una licencia Unix. Significa que se evitará mucho del derroche de esfuerzos en la duplicación de la programación de sistemas. Este esfuerzo puede enfocarse mejor para hacer avanzar el estado de la técnica. (Stallman 2004, p. 38).

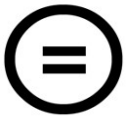
A finales de los noventa el término *software libre*, que llevaba ya varios años de existencia, comenzó a tener una gran exposición con el auge del Internet. Desde diversos campos se empezó a analizar el impacto legal y económico del llamado *free software* como un mecanismo capaz de producir alternativas a las vías ya establecidas en el ámbito informático, como el software propietario. Estas alternativas han seguido surgiendo y se han visto apoyadas por diversos mecanismos como *Creative Commons* creado a principios del siglo XXI.

> La idea de Creative Commons es contribuir a la creación de un espacio que promueva, facilite y garantice el intercambio colectivo de obras y trabajos de artistas, cientistas (*sic*) y diferentes desarrolladores, como forma de proteger una cultura diversa, de libre intercambio y que defienda la libertad de expresión. (Vercelli, 2004, p. 152).

A continuación se presenta un cuadro que explica las licencias de Creative Commons[8], extraído del texto de Ariel Vercelli, "La conquista silenciosa del ciberespacio" (2004, p. 158):

	"Attribution: The licensor permits others to copy, distribute, display, and perform the work. In return, licensees must give the original author credit."
	"Atribución: El licenciante permite a otros copiar, distribuir, exhibir y ejecutar la

[8] La guía completa sobre las licencias puede consultarse en http://creativecommons.org/licenses/by-nc/2.0/

	obra. A cambio, los licenciatarios deben dar el crédito al autor original.” (Se permiten trabajos derivados y usos comerciales).
	“Noncommercial: The licensor permits others to copy, distribute, display, and perform the work. In return, licensees may not use the work for commercial purposes -- unless they get the licensor's permission.”
	“No comercial: El licenciante permite a otros copiar, distribuir, exhibir y ejecutar la obra. A cambio, los licenciatarios no pueden usar la obra con fines comerciales -- a menos que consigan el permiso del licenciante.”
	“No Derivative Works: The licensor permits others to copy, distribute, display and perform only unaltered copies of the work -- not derivative works based on it.”
	“Sin Trabajos Derivados: El licenciante permite a otros copiar, distribuir, exhibir y ejecutar solo copias inalteradas de la obra – no trabajos derivados basados en este.”

	“Share Alike: The licensor permits others to distribute derivative works only under a license identical to the one that governs the licensor's work.”
	“Compartir en Idénticas Condiciones: El licenciante permite a otros distribuir trabajos derivados solo bajo una licencia idéntica a la que gobierna su obra licenciada.”

2.2.1 Características del Software Libre

Se conoce como Software libre a la creación cooperativa de código para un programa y su posterior distribución de forma abierta y libre, a través del uso de diversas licencias distintas al *copyright*. Esta práctica “ejemplifica una considerable reorientación del conocimiento y el poder en la sociedad contemporánea”. (Kelty, 2008, p. 17) Como práctica cultural abarca diversas regiones geográficas y perfiles profesionales; los grupos (académicos, músicos, etcétera), así como la intención particular de cada programa que se distribuye como software libre, son muy diversos. Las personas que lo escriben no lo hacen con los mismos objetivos, pero ayudan a crear una comunidad que comparte un implícito imaginario social que encuentra una relación ideológica política; en algunos casos es una visión anti-mercantilista, y todos los proyectos funcionan con el ideal de abrir el código de su software y compartirlo con la intención de que esta comunidad lo mejore. El software libre, más que una ideología es una práctica, y el usuario es el mismo encargado de preservarlo y extenderlo.

El término *free* hace referencia al significado de libertad, aunque en español no hay tanto problema como con *free* en inglés (se puede entender como gratuito), se ha provocado un malentendido popular respecto a que la idea del software libre es ser un producto gratuito, y si bien, habitualmente el software libre se distribuye sin costo, no es una obligación que sea de este modo; muchos proyectos reciben dinero, la mayoría en forma de donaciones para sostenerse. De este modo, más allá de la cuestión económica, lo que define al software libre es la apertura de su código fuente.

Esta práctica, que revela la importancia del código fuente, favorece que un programa pueda desarrollarse y mejorarse rápidamente. El código fuente de un programa es parte de las tecnologías que hacen funcionar a las computadoras, “es el conjunto de instrucciones,

primero escritas como palabras, que dirigen la funcionalidad de las máquinas". (Stallman, 2004, p. 11) Es escrito por programadores, es decir, un actor humano que indica las instrucciones de funcionamiento legibles en códigos procesables para las computadoras o actores-no-humanos; el código regula aquello que Lessig (1999, citado en Vercelli, 2004, p. 7) llama el espacio de las aplicaciones de Internet; es evidente su importancia, y por ende, las bondades que ocasiona que sea accesible para todo aquel que lo desee.

Como un par de características esenciales del software libre, se pueden mencionar la reutilización y modificación. La posibilidad de modificar el código permite, por ejemplo, que la infraestructura de un programa perdure, ya que aunque se modifique el hardware con el que trabaja el software se puede adaptar con relativa facilidad. La reutilización funciona en el sentido de transformar el uso que se le dio a una idea programacional hacia un objetivo distinto pero también con la intención de mejorar y redistribuir el programa original.

Para tener una mayor comprensión del impacto cultural del software libre se puede retomar el trabajo de Christopher Kelty, (2008) quien habla de tres fenómenos que están íntimamente relacionados: *Internet* refiriéndose a la infraestructura tecnológica; el *software* libre, como un conjunto de prácticas legales, sociales y técnicas; y los *públicos recursivos,* un concepto analítico que pretende aclarar la relación entre internet y software libre. Internet y software libre son conceptos que se pueden ubicar históricamente, no así públicos recursivos.

> [De acuerdo con Kelty] el software libre se entiende como "un conjunto de prácticas para la creación cooperativa y distribuida de código fuente, el cual luego se difunde de forma abierta y libre a través de un uso astuto e insólito de la legislación de copyright" pero es mucho más que eso, el software libre ejemplifica una considerable reorientación del conocimiento y el poder en la sociedad contemporánea —una reorientación del poder con respecto a la creación, diseminación y autorización del conocimiento en la era de Internet. (Kelty, 2008, p. 17).

Por otro lado define como público recursivo aquel que está comprometido con la conservación y modificación material y práctica de los medios técnicos, legales, prácticos y conceptuales de su propia existencia como público. Se diferencia de un público convencional que sólo percibe a uno que realiza acción directa, al formar parte de la infraestructura de aquello que siguen, no es un grupo independiente de las estructuras de poder o

gubernamentales, más bien es autónomo respecto a los mecanismos que busca, en la práctica se podría decir que el software libre es *para* y *por* una comunidad.

Kelty considera que Internet es un espacio que permite florecer la práctica del software libre y la comunidad de público recursivo alrededor de él.

Ahondando en el concepto de Internet, se puede utilizar el concepto que dio Lawrence Lessig, (2001, citado en Vercelli, 2004, p. 42) mencionando que Internet se divide analíticamente en tres capas: la primera de ellas, en orden ascendente, es la capa de la infraestructura (computadoras, satélites, cables y hardware); "[l]e sigue la capa lógica, o capa del código, aquella capa intermedia que concentra los componentes más blandos, o sea, el software y las aplicaciones que hacen funcionar una red de redes como Internet", (Piscitelli, 2005, p.18) históricamente esta capa fue abierta, transparente y flexible, pero desde la década de 1980 ha encontrado cambios dirigidos a un cierre parcial y generar un sistema de códigos abiertos versus códigos cerrados; y finalmente, la capa superior de Internet, que está compuesta por los contenidos que circulan. Esta última capa de contenidos, a diferencia de la capa lógica, ha comenzado un proceso de privatización desde mediados de la década de 1990, y es donde el acceso a los contenidos musicales ha generado controversia desde la aparición de las tecnologías *peer to peer*.

> Lo que se promociona como lucha contra la piratería o defensa de los derechos de las discográficas, en rigor no es sino una discusión mucho más profunda acerca del acceso a la información como derecho universal, con múltiples variantes y consecuencias. (Piscitelli, 2005, p. 17).

El software libre constituye un ejemplo de cómo conceptualizar esa capa de contenidos, desde su aparición en el pasado reciente y la dirección que debe seguir en el futuro. Se convierte en algo que trasciende una cuestión técnica frente al control de una corporación o un gobierno; es propiamente una comunidad que realza las características naturales de Internet y que pretende rebasar la intención comercial, buscando garantizar un acceso democrático a la información en la red. Esto sucede gracias a que son estos mismos principios los que técnicamente hicieron posible la conformación del Internet, esta arquitectura abierta permitió el acceso público y sin restricciones, como una plataforma (y esto es especialmente importante para este trabajo) en la que cualquier persona con los

recursos técnicos disponibles ha podido desarrollar y transmitir su obra, código o herramientas de *software*.

Internet es el nuevo medio de divulgación científica, por ende, si la ciencia debe divulgar el conocimiento e Internet es el medio contemporáneo, la información y el conocimiento a través de Internet debe ser gratuito. No se pueden generar mecanismos que obstruyan el progreso de la ciencia, como lo sería evitar la distribución del código.

2.3 Software Libre en el arte

El creador artístico, al registrar proyectos como software libre utilizando licencias como Creative Commons, cumple una función social como artista, pretende la creación colectiva y autónoma de objetos técnicos complejos que son distribuidos de forma libre y pública. Este concepto de propiedad intelectual se fundamenta en una visión opuesta a la que el entorno mercantil ha creado, donde el deseo de ser recompensado por la creatividad no justifica privar al mundo del producto de esa creatividad. La intención del *free software* respecto a democratizar el conocimiento favorece la investigación científica y el desarrollo tecnológico vinculado al desarrollo social, “es posible que nos encontremos frente a la gestación de un nuevo movimiento social global que se aboca a la protección de la diversidad cultural, el dominio público y los valores que sustentan las sociedades abiertas y libres.” (Vercelli, 2004, p. 150) En este sentido, Internet necesita estos mecanismos porque de no ser así puede transformarse en un espacio de control absoluto cuyas regulaciones busquen la estabilidad de un mecanismo mercantil que beneficie a empresas y no a la población ni a la ciencia. Para el avance informático de nuestros días, es necesario que como actores dentro del sistema de comunicación, los usuarios del software ayudemos a alcanzar una regulación de Internet justa y orientada al beneficio de la población general. Son necesarias regulaciones que impidan el uso restringido de la red, restricción que ya ha iniciado y que Lessig destaca en el subtítulo de su trabajo *Cultura Libre*: (2005) “Cómo los grandes medios están usando la tecnología y las leyes para encerrar la cultura y controlar la creatividad”, reflexión que incluye la Web 2.0, que intenta ser limitada por empresas que se dedican a la difusión pero no a la producción de nuevos contenidos culturales, como puede ser el caso de distribuidores de contenido audiovisual.

> Según Kevin Kelly editor ejecutivo de la revista Wired, la Web 2.0 es el nuevo sistema operativo social en el cual emerge un "diseño espacial en que la coordinación pública descentralizada puede resolver los problemas y crear cosas que ni el comunismo puro ni el capitalismo puro pueden" (Kelly, 2009). Se trata de una nueva inteligencia colectiva que "se transforma a la luz de la conectividad" que permite que los usuarios intercambien información y creen no sólo conocimiento y gestionen contenidos en sitios como Digg o de.licio.us, sino que se logren nuevos mecanismos de préstamo en sitios como Zopa o Wesabe que eluden la intermediación bancaria. Semejantemente, listas como OLX y Craigslist eliminan la intermediación comercial al facilitar intercambios individuales. (Yudice, 2009, p. 93).

La arquitectura de Internet originalmente está diseñada para el acceso y la proliferación de cualquier tipo de manifestación, incluso las contraculturales. Esta libertad natural y original de Internet puede ser afectada si el motor para realizar programas es meramente mercantil y de carácter privativo, ya que a su vez puede producir una red con un entorno altamente regulable mediante un control central. "A diferencia de lo que ocurría hasta hace una década, la proliferación de dispositivos de control puede consolidar arquitecturas irreversibles que privilegien el control y la vigilancia, a la diversidad y libertad."(Vercelli 2004, p. 55)

Dentro de las características más perjudiciales de esta postura de software propietario, de uso reestringido y cuya estructura no es abierta al público, es que un menor número de personas usa el programa tanto por el filtro económico como por el que ningún usuario puede adaptar o arreglar el programa. Bajo esta segunda manera, otros desarrolladores no lo pueden mejorar o basar un trabajo nuevo en él.

El software se ha convertido en una herramienta e incluso en el medio para desarrollar distintas actividades en la actualidad, esto ha sido explotado por empresas, que incluso han formado monopolios informáticos. Esta especie de tiranía trasciende la cuestión mercantil, pues pretende voluntaria o involuntariamente establecer patrones (a veces no modificables) que el usuario debe seguir en su producción, así tenemos por ejemplo *photoshopear* como un término que a menudo se entiende como editar imágenes fotográficas. En el campo del software para creación artística, el modelo comercial crea programas perfectamente funcionales pero que no permiten darles otro uso o reconfigurarlos, no hay cambios permisibles, la interfaz rara vez es tan modificable como para satisfacer las expectativas de usuarios con diversos intereses.

La facilidad de tener *Home Studios* ha democratizado las tecnologías musicales, pero

el modelo comercial a menudo está basado en el paradigma de los estudios analógicos. En algunos casos se intenta imitar la forma, aunque no necesariamente retoma todas las ventajas de la música por computadora, principalmente la programación, esto puede ir limitando incluso la creatividad, pues se quiere establecer uniformidad en el flujo de trabajo creando un público verdaderamente cautivo.

En términos teóricos este tipo de cuestiones derivadas del software propietario nos remite al concepto que Theodore Adorno definía como Industria Cultural. (Horkheimer y Adorno, 1988) Entendida ésta como las técnicas de reproducción industrial en la creación y difusión masiva de obras culturales. Para Adorno, los procedimientos técnicos de producción de la industria cultural apuntan a la estandarización y la uniformidad de sus productos, a la producción en serie y la distribución en masa. (Zamora, 2001)

El artista basado en su especialización debe reaccionar para construir mecanismos que resistan esa perspectiva. “El papel del artista frente a la tecnología muchas veces es el de subvertir los usos preestablecidos, encontrando nuevas interpretaciones de herramientas existentes”. (Machado, 2006, p. 146) Así, al construir herramientas artísticas por medio del software libre y la colaboración de comunidades *online* en las mismas, se adopta una postura de reacción por parte del artista a la imposición que resulta de la industria cultural en términos de Adorno, debido a que el proceso creativo deja de ser delimitado por estándares dictados por empresas de software, aquí radica una de las mayores importancias del software libre devolver esa libertad al usuario, lo que recuerda, quizás de forma fugaz, a cuando el mismo Adorno mencionó que Arnold Schönberg proponía una ruptura al estándar de la música tonal a través de su sistema dodecafónico, cuyo planteo vanguardista abrió el camino al serialismo y otras perspectivas.

La industria cultural, por medio de moldes, construye productos individuales. Una alternativa a ésta, es que cada quien modifique esos moldes de acuerdo a sus necesidades. No podemos olvidar que vivimos “en un mundo dominado por las concepciones de distintos universalismos, de la tonta presunción humana de poseer la certeza universal.” (Beck, 1998: 126) Dejar de lado los metarrelatos sitúa a la ciencia, la cultura y el arte en un lugar distinto, pues se afecta el saber mediante la modificación de la investigación y la transmisión de información.

Las nuevas tecnologías permiten que esto suceda de esta manera. Las prácticas son

diversas y novedosas, por ejemplo, el documental *RIP: manifesto remix*, acerca de los cambios en el concepto de derechos de autor, cuya peculiaridad reside en que es un film de código abierto, es decir se puede descargar de forma libre, editar y subir de nuevo a la red para generar y compartir una versión propia del proyecto, algo totalmente contrario a los productos homogeneizados y estandarizados que se crean en la industria cultural cuya principal intención es que el receptor consuma, no que produzca.

En el caso específico del desarrollo de herramientas artísticas, la oportunidad de diseñar aplicaciones personalizadas y aprovechar el avance que logra una comunidad, hace que el software libre permita crear a cada uno sus propias librerías e interfaces. Actualmente existen herramientas que han demostrado la eficiencia de este modelo. Tanto en la distribución como en la construcción de las mismas, el tiempo de trabajo para alcanzar la madurez de los programas, así como las posibilidades de expansión que ofrecen, han mejorado exponencialmente.

El software libre como una herramienta que puede ser modificada, re-creada y manipulada ayuda a potenciar el proceso creativo del artista que utiliza herramientas digitales. Sin embargo, su aporte también incluye un modelo colaborativo de producción que puede trasladarse a otros ámbitos de la creación artística. “El software y en particular el libre, aparece como medio, proceso y producto transversal a las prácticas artísticas”, (Medina, 2010) convirtiendo el proceso de construcción de las herramientas en un acto de creación colectiva asociado al arte, mediado por las nuevas tecnologías, utilizando propuestas estéticas como inspiración para la invención de recursos tecnológicos. En este sentido, el software es un elemento polisémico, producto de varios autores, que rescata la estética de la creación colectiva y la reinterpretación de contenidos, bajo una ética de acceso libre.

Entender el software libre como un recurso público es poner importancia en su lado de significado democrático y social o político. Poner atención en ese rubro es también el papel del creador artístico y de la universidad. “Un artista es un investigador, siempre empujando las fronteras de lo posible. El artista trabaja para cambiar nuestra percepción del mundo, indaga en el medio y las técnicas que ha elegido, experimentando y desarrollando nuevos caminos y visiones, no importa si es pintura, piedra o código, el proceso es el mismo”. (Soler, 2008, p. 14)

La fusión arte-ciencia a través de la construcción de herramientas artísticas por medio

del software libre y la colaboración de comunidades *online,* da también otro matiz a la relación arte-sociedad: rechaza el conformismo y la imposición que resulta de la industria cultural que mencionaba Adorno. “La democratización del conocimiento apunta a revalorizar la práctica de la investigación científica y tecnológica en vinculación con los objetivos del desarrollo social” (Rietti y Maffía, 2005 citado en Massarini, 2007) con la intención de resolver la dinámica de producción de bienes y servicios y la problemática social.

El arte, en su papel dentro de la sociedad, es un servicio público; el músico, como artista también, tiene una función social como creador de arte, de ahí la importancia de mostrar que la música permite esquemas funcionales que se pueden trasladar a otras esferas y así contribuir al mejoramiento de mecanismos dentro de la sociedad.

> La música tiene pues una naturaleza auténtica determinada por los medios sociales, culturales y económicos, está sujeta a los cambios individuales y está bullendo en relaciones de significación que dependen de una enorme multitud de factores. Por ello su función primera no debe ser buscada en la estética abstracta, sino en la eficacia de su participación en una regulación social históricamente determinada.(García, 2002, p. 158).

El *free software* como se le conoce en inglés al software libre significa entonces una gran comunidad de programadores y usuarios en busca de que un determinado programa se difunda, y si es posible, evolucione rápidamente. Así mismo, es importante recalcar que esto significa que las diversas herramientas son producto del trabajo de muchos programadores los cuales colaboraron para poder ejecutar el resultado final.

El que la mayoría de la población tenga acceso a los ordenadores hace que las formas en cómo pueden afectar las dinámicas del software libre a la música, abran paso al siguiente capítulo. Sin olvidar que el propósito, puede ser estandarizar o liberar un programa, depende como siempre de la intención del creador, no de la herramienta. A final de cuentas las computadoras no fueron diseñadas para algo en específico, sino para realizar cálculos exactos.

Es importante mencionar que en la actualidad el software propietario posee características como la estabilidad y amabilidad con el usuario que no siempre son replicadas por sus pares libres. De hecho, dependiendo las necesidades y sobre todo la familiaridad del usuario con la práctica informática, en muchos casos el software comercial

cubre de forma eficiente las necesidades que puede buscar el consumidor, por lo que es evidente que en el futuro inmediato (quizás permanentemente) ambas posturas convivirán en el universo informático, siendo principalmente la postura del presente trabajo la de entender el software libre como una alternativa, más que como la solución a todo lo que pueda requerir un usuario.

Capítulo III

Laptop performance mediante *SuperCollider*

Trabajar con sonido por medio de una computadora es una de las opciones más atractivas dentro de los usos que se le pueden dar a un ordenador, dada su precisión y las posibilidades que ofrece que son de gran utilidad.

En este sentido, a diferencia de los instrumentos acústicos que tienen un determinado timbre, teóricamente la computadora podría generar cualquier tipo de sonido utilizando diversos valores y operaciones matemáticas, cuyas combinaciones ofrecen posibilidades prácticamente infinitas, aunque las computadoras, como herramientas, también son completamente inútiles si no tienen las instrucciones adecuadas para realizar una tarea. Es decir, necesitan que se les diga exactamente qué hacer y cómo hacerlo. Para esta interacción se requiere de un programa que proporcione la secuencia de instrucciones precisas que debe seguir una computadora para funcionar, dichas instrucciones están escritas en lo que se conoce como lenguajes de programación, una colección de sintaxis y reglas para especificar los pasos lógicos durante un proceso computacional, y así traducir el lenguaje humano a un lenguaje que la computadora comprenda. En otras palabras, los lenguajes de programación no ejecutan en sí mismos una tarea específica sino que permiten construir un algoritmo o un programa que la ejecutará.

Hoy en día la programación es una disciplina en expansión gracias al uso común de la computadora. Existe una gran variedad de lenguajes de programación, desde el llamado lenguaje ensamblador que es en el que está programada la computadora en su nivel más básico, hasta lenguajes de muy alto nivel, mucho más similares al lenguaje humano. Actualmente también hay entornos de programación que permiten simular sonidos, imágenes u objetos tridimensionales, los cuales pueden ser utilizados para la creación artística en todas sus ramas y ha derivado en innumerables direcciones, estilos y movimientos que forman parte del arte digital.

De esta forma, si estos lenguajes son distribuidos bajo las premisas del software libre, se habla de públicos recursivos, concepto mencionado en el capítulo anterior y que da origen a una comunidad de músicos usuarios de computadora que constantemente se benefician entre sí al compartir herramientas, librerías y archivos que pueden contener incluso las

instrucciones detalladas para generar un sonido específico.

Quizás esa oportunidad para los artistas de conectarse entre sí, intercambiando libremente las ideas, fundamenta la importancia de esta perspectiva.

3.1 La programación como herramienta en la música

Generar sonido por medio de un lenguaje de programación implica, igual que otros procesos de creación, ir detallando paso a paso los eventos necesarios para conseguir el objetivo deseado. De manera similar sucede en el caso de la música en los procesos que sigue un compositor al crear una estructura armónica o un artista sonoro al buscar algún sonido específico.

Es importante mencionar que crear música mediante programación es una de las mejores formas para adentrarse en este terreno. Existen diversas posibilidades en la producción musical a través de la programación, que pueden ir desde la ejecución de un determinado algoritmo que ejecute la estructura de una composición, a la construcción de un motor de síntesis digital. “El uso de computadoras en los procesos creativos orilla a que pensemos la comunicación y los procesos creativos en términos de estructuras abstractas y en la manipulación de este tipo de estructuras.” (Holtzman, 1994, citado por Sorensen, 2005, p. 1)[9]

Tomando en cuenta esas posibilidades, entendemos que por supuesto no todas las ideas musicales pueden adaptarse a un proyecto de programación, o que por lo menos, no resulta práctico en todos los casos. La intención de crear sonido por medio de una computadora no se limita a emular las prácticas musicales tradicionales. En este sentido la computadora y el software funcionan también como un instrumento para la ejecución musical en directo, lo cual ha abierto un campo de desarrollo para la música en vivo, el *laptop performance*, concepto que retomo de Collins (2003) y Zadel (2006), y que se entiende como la interpretación de música en directo usando para su ejecución computadoras portátiles, con la característica distintiva del uso directo de las herramientas de software para las presentaciones.

El uso de *laptops* en el escenario empezó a proliferar hace poco más de diez años. Si

[9]Original en inglés: “The use of computers in the creative process mandates that we think of communicative and creative processes in terms of abstract structures and the manipulation of such structures.”

bien, regularmente los programas más usados para estas presentaciones son de software propietario, es evidente que no todos los usuarios encuentran satisfacción en el uso de programas como *Ableton Live* o *Reason* cuyas interfaces gráficas y paradigmas de trabajo son rígidos, por lo que lenguajes de programación principalmente bajo el paradigma del software libre se han establecido como una alternativa, que además de posicionarse bajo otro flujo de trabajo, suelen permitir la creación personalizada de interfaces. En el presente trabajo, nos enfocaremos en esta segunda fórmula del *laptop performance*.

Si bien el uso de estos programas para hacer música aún dista mucho de ser popular o de competir con el software propietario, es previsible una expansión de usuarios, tomando en cuenta que para las generaciones venideras, programar será una habilidad de uso común, y aprendida en edades más tempranas cada vez. Por supuesto, no está de más aclarar que esta forma de trabajo sirve como una herramienta musical, que como todas las que tengan que ver con una computadora, ofrece ventajas y desventajas con respecto al proceso de creación.

3.2 El tiempo real en el laptop performance

“Give us access to the performer's mind,
to the whole human instrument."
(www.toplap.org)

Los primeros ambientes de programación que producían sonido implementaban un método de trabajo por lotes: edición-compilación-ejecución, sin embargo, el paradigma de programación que permite compilar y ejecutar en un mismo paso y posteriormente reevaluar se ha establecido como una alternativa, lo que abre la posibilidad de utilizar herramientas dinámicas en las actuaciones en vivo.

La creación y manipulación de algoritmos para crear sonido en tiempo real mantiene un emparejamiento evidente con la programación interactiva que consiste en modificar programas mientras se ejecutan, en lugar de escribir, compilar y después probar el código, algo que permiten ambientes de desarrollo en Lisp o Haskell, por ejemplo, y se ha convertido

en un camino que el usuario puede utilizar para sus presentaciones.

En un principio, en éstas se realizaban procedimientos imperceptibles para el público lo que llevó al *laptop performance* a incluir músicos y también algunos otros elementos visuales para hacer más atractiva las presentaciones. Sin embargo, dentro de la programación en tiempo real, la práctica que muestra el proceso de programar ha adquirido relevancia en círculos especializados, el *Live Coding*, como se le llama, tiene como peculiaridad mostrar a los miembros de la audiencia el proceso de creación de la obra, lo que permite comprender a aquellos familiarizados con las estructuras y las abstracciones del código la forma en la que se está creando la pieza.

Esto ofrece un entorno novedoso para la exploración de la estructura del arte, y al resto de la audiencia le proporciona una experiencia más cercana con el artista, pues el *live coding* proporciona esa sensación escénica que las presentaciones de músicos por computadora no solían ofrecer. Como todo arte, una gran presentación al utilizar programación lleva consigo una habilidad muy difícil de alcanzar, que requiere tiempo, dedicación y talento. Si la práctica como tal prospera lo suficiente, se podrán ver presentaciones técnicamente muy virtuosas.

Al respecto, es importante mencionar la existencia de la organización TOPLAP, que promueve esta práctica. Su sitio web[10] ofrece datos sobre lo que ellos consideran antecedentes históricos además de agrupar artículos académicos referentes al tema, una lista de artistas e incluso un manifiesto.

Según TOPLAP, el *live coding* tiene su antecedente en los *happenings* de los 60's-70's, y es en 1985 que se realiza la primer presentación de *live coding* que se tiene documentada en un concierto realizado por Ron Kuivila en el STEIM (Studio for Electro Instrumental Music) de Amsterdam en 1985, utilizando un lenguaje basado en Forth, el cual, junto con Lisp, sirvieron durante la década de los 80 y principios de los 90 para ese tipo de presentaciones.

Con la excepción de pequeños focos de actividad el live coding parece haber estado ausente durante los años 90. La programación en vivo durante este periodo parece haber sido limitado a la manipulación de las rutas de señal en patches de Max. Un resurgimiento en el uso de los lenguajes basados en texto para la programación en vivo se pudo observar en

[10]http://toplap.org/index.php

el año 2000 con los experimentos de Julián Rohrhuber en Supercollider. (Sorensen, 2005) También es posible encontrar a principios de este siglo trabajos importantes relacionados con el *live coding* por parte de Alex McClean, Adrian Ward y Nick Collins, así como la creación del lenguaje ChucK diseñado por Ge Wang y Perry Cook.

El *live coding* ha sido difundido en los diversos congresos de música por computadora y a pesar de ser aún reciente, vislumbra grandes posibilidades mediante el desarrollo de sus herramientas y técnicas. El uso generalizado de la computadora y el conocimiento cada vez más común del proceso de programación, posibilita el interés del creador musical en una práctica que utiliza código textual para realizar síntesis de sonido, tanto por la simplicidad de ejecución como por las posibilidades creativas que ofrece.

A continuación una tabla sobre pros y contras del *live coding,* traducido de Collins, N., McLean, A., Rohrhuber, J. & Ward, A. (2003).

Ventajas y potencial	Desventajas y peligros
Flexibilidad; la siguiente sección puede ser cualquier cosa	Puedes olvidar quitar lo que suena en el momento o simplemente tomar demasiado tiempo mientras se prepara la siguiente sección
Un gran desafío intelectual	
Cambios arbitrarios y complejas en la estructura durante la presentación	Tu concentración se divide entre la adrenalina / estrés / cerveza en un concierto
Una nueva forma de improvisación	Es arriesgado simplemente correr código! Si depuración o pruebas disponibles
Programas normales para presentaciones en directo como Reason lucen aburridos si se proyectan a una pantalla, pero los misteriosos sistemas de codificación de texto tienen su encanto	Debes aceptar algunas fallas y momentos desagradables en el sonido, hay una compensación entre la preparación y la especificidad del concierto
Es gratificante ver el esfuerzo real traducido al sonido	La ofuscación deliberada no es un gran criterio para el arte!
Los lenguajes de programación son inmensamente ricos en su gramática	Se aplica mucho esfuerzo para una pequeña recompensa y se debe tener código de respaldo en caso de que no llegue la inspiración.
	Escribir en una computadora no es el método de interconexión más emocionante visualmente hablando – estarás inclinado

	sobre una pantalla toda la noche a menos que programes desde controladores externos

Diversas universidades que manejan la “Tecnología Musical” en sus programas de estudio tienen investigaciones que de una u otra forma se relacionan con ambientes de programación basados en el control por medio del texto para crear música en tiempo real y han derivado en proyectos ya sea personales o institucionales, cuyo resultado es ambientes y programas que sirven para *live coding.* Para este trabajo, el más importante y sobre el que se ahondará en el siguiente apartado es *SuperCollider*[11] un ambiente de programación creado por James Mcartney y basado en lenguajes como *C* y *SmallTalk*, diseñado para síntesis de sonido y composición algorítmica así como el trabajo en tiempo real. Además de *SuperCollider* hay otros programas que se han difundido entre las personas interesadas en el *live coding*, como por ejemplo *ChucK*[12] creado por Ge Wang y Perry Cook, en la Universidad de Princeton como parte del proyecto *On-the-fly Programming* en el que se refieren al código como un expresivo instrumento musical, también existen entornos como *Impromptu*[13] y *Overtone*[14] que integran además del audio una fuerte interacción con ambientes diseñados para la creación de visuales en tiempo real, o lenguajes completamente diseñados para el *live coding, Tidal*[15] en el caso de la música y ejemplos como *Fluxus*[16] y *Gamuza*[17] más orientados a la parte visual. Finalmente, no se debe dejar de mencionar ambientes de programación que no están basados en el control de texto sino que utilizan entornos gráficos para realizar *live coding,* como lo son *MAX/MSP*[18], *Pure Data*[19] y *VVVV*[20], este último más enfocado a la generación de visuales pero con prestaciones para la parte sonora.

La creación de código en tiempo real va encontrando cada vez un mayor interés, existen un par de cortos documentales sobre el fenómeno *Algorithms are Thoughts,*

[11] Referencia para descarga: http://supercollider.sourceforge.net/

[12] Referencias: http://on-the-fly.cs.princeton.edu/ y http://chuck.cs.princeton.edu/

[13] Referencia: http://impromptu.moso.com.au/

[14] Referencia: http://overtone.github.com/index.html

[15] Referencia: http://yaxu.org/tidal/

[16] Referencia: http://www.pawfal.org/fluxus/

[17] Referencia: http://gamuza.d3cod3.org/

[18] Referencia: http://cycling74.com/downloads/

[19] Referencia: http://puredata.info/

[20] Referencia: http://vvvv.org/

Chainsaws are Tools[21] de Stephen Ramsay y *Show us your screens*[22] de Louis McCallum and Davy Smith, en México, además de las sesiones de *Live Coding* organizadas por parte del Centro Multimedia del CENART y de diversos conciertos a lo largo de recintos culturales y universidades, se organiza desde 2012 el Simposio Internacional de Música y Código llamado */*vivo*/*[23].

3.3 SuperCollider

SuperCollider es un ambiente de programación apto para *laptop performance,* combina la programación orientada a objetos y la programación funcional, utiliza elementos de sintaxis del lenguaje C, sin embargo la base más importante sobre la que está diseñado es un lenguaje llamado *SmallTalk*.

> SuperCollider está basado en muchos lenguajes de programación, pero el lenguaje del que pide prestado elementos con más fuerza es uno llamado Smalltalk. Smalltalk, como SuperCollider, es un lenguaje orientado a objetos. Cuando tomé el curso de Lenguajes de programación en la universidad, mi profesor dijo que Smalltalk fue el mejor lenguaje orientado a objetos y la única razón por la que no era el más popular es que la sintaxis era una locura. (Hutchins, 2005,p.4)[24].

SmallTalk fue creado a partir de investigaciones coordinadas por Alan Kay en el Xerox PARC durante los años setenta. Mientras buscaban la creación de un sistema informático orientado a la educación, crearon un dispositivo llamado *Dynabook*, el primer intento de una tableta digital pensada para que lo usaran niños, el componente de software de esta investigación fue *SmallTalk*.

SmallTalk contó siempre con una interfaz gráfica y fue fundamental para generar el paradigma de orientación a objetos, en el cual existen objetos que reciben mensajes. La sintaxis de SmallTalk es una consecuencia de eso:

---

21 Referencia: https://vimeo.com/9790850
22 Referencia: https://vimeo.com/20241649
23 Referencia: http://vivo2013.cenart.tv/index.html
24 Original en inglés: “SuperCollider is based on many other programming languages, but the language that it borrows most heavily on is one called Smalltalk. Smalltalk, like SuperCollider, is an object-oriented language. When I took Programming Languages at uni, my teacher said that Smalltalk was the best object oriented language and the only reason it wasn't the most popular was that the syntax was insane.”

Objeto-mensaje

Todo en SmallTalk es un objeto, mientras una clase describe la implementación de un conjunto de objetos y un mensaje es un requerimiento que se le hace a los mismos. Así, los métodos son la forma de especificar cómo los objetos de una determinada clase responderá a los mensajes. La estructura interna de los objetos está compuesta por variables de instancia.

SmallTalk es un lenguaje de programación caracterizado por poseer encapsulación, polimorfismo y herencia, pero es también una librería de clases y un entorno de desarrollo.

> SmallTalk, además de ser un lenguaje de computación muy poderoso y versátil, es una interpretación,de como debieran utilizarse las computadoras: Las computadoras deberían ser herramientas que sirvan para amplificar el espíritu creativo de las personas...
> Se promueve un método de desarrollo donde el software cambia conforme las personas que lo desarrollan lo hacen. (Gomez 2006, p. 7).

En este sentido, *SmallTalk* no es el fin de este proceso, sino un paso más, el objetivo como tal es diseñar mediante *SmallTalk* y que la herramienta permita volver obsoleto al mismo proceso. Es decir, una ampliación de las características mediante la misma herramienta, a pesar de que no sea muy difundido el discurso que existe alrededor de *SmallTalk*. Una herramienta como *SuperCollider* continúa con esta postura permitiendo diseñar herramientas personalizadas que permiten realizar tareas específicas y así expandir el programa y en algún punto incluso sustituirlo. *SuperCollider* es un ambiente de programación para síntesis y creación de sonido que se utiliza también para composición algorítmica y ejecución musical en tiempo real, es un programa útil para la creación de música generativa e interactiva, en tiempo diferido y en tiempo real.

> La idea detrás de SuperCollider era proporcionar un motor de síntesis de audio en un lenguaje de alto nivel con tipado dinámico, listas, recolección de basura y cierres. Normalmente, un lenguaje de este tipo es demasiado lento para realizar procesamiento de señales. Es posible amortizar el costo computacional de usar muchas muestras de audio mediante la creación de un tipo de señal de datos que representa un tampón de muestras durante un corto período de tiempo, así como definiendo los operadores para ese tipo de datos. (McCartney, 1996)[25].

[25] Original en inglés: ”The idea behind SuperCollider was to provide audio synthesis in a high level language with dynamic typing, lists, garbage collection and closures. Normally a language of this type is too slow to do

SuperCollider cuenta con un intérprete o cliente (sclang) y un servidor (scsynth) que se comunican mediante el protocolo OSC. A partir de la versión 3 de *SuperCollider*, la cual se hizo de código abierto, el servidor del programa permite que cualquier lenguaje de programación pueda controlar el motor de síntesis, por lo que abrió las puertas a utilizar el poder de *SuperCollider* con el lenguaje de script que cada usuario prefiera en lugar de forzar al usuario a utilizar la sintaxis de *SmallTalk*, ya que comparte formas con muchos lenguajes orientados a objetos. "Las motivaciones para el diseño de Supercollider fueron: la capacidad de darse cuenta que los procesos de sonido eran diferentes cada vez que se ejecutaban, escribir piezas que describieran la gama de posibilidades más que entidades fijas y facilitar la improvisación en vivo para un compositor/intérprete." (Amatriain, 2004, p. 128)[26]

El usuario puede escribir algoritmos para síntesis y para composición en el mismo nivel de lenguaje, lo que permite la creación de instrumentos virtuales de una forma flexible. En SuperCollider, los bloques de código pueden ser evaluados de uno en uno, y las funciones, números o instancias de objetos pueden ser asignados a variables, por ejemplo:

a = 10 (las variables que se identifican con una letra son globales y no necesitan declararse)

A la vez, éste es un programa que funciona didácticamente para que las personas entiendan y diseñen algoritmos que se ajusten a sus necesidades exactas ya que permite crear páneles de control y pantallas gráficas. *SuperCollider* es adecuado como herramienta para la enseñanza de diversas técnicas de síntesis; se distribuye como software libre lo cual ha servido para difundirse ampliamente y ha consolidado una comunidad de usuarios muy activa que permite una interacción, retroalimentación y aprendizaje continuo sobre las utilidades del programa.

Una de las prestaciones más importantes del programa es que los usuarios pueden compilar nuevos bloques de código mientras se ejecutan bloques ya compilados. En este sentido *SuperCollider* incluye librerías y extensiones que permiten la ejecución musical en tiempo real, en particular, la librería llamada *JITLib*. La cuál se incluyó como parte de

signal processing. By creating a signal data type which represents a buffer of samples over a short time frame, and defining operators for that data type, it is possible to amortize the cost of the interpreter over many samples."

[26] Original en inglés: "Motivations for the design of Supercollider were the ability to realize sound processes that were different every time they played, write pieces describing ranges of possibilities rather than fixed entities and to facilitate live improvisation by a composer/performer."

SuperCollider desde la versión 3.

> Mi razón para introducir la librería *just in time programming* (jitlib) era crear una interfaz para escribir código durante una presentación en vivo, lo que elimina la distinción entre preparación y acción, para que yo pudiera cambiar más fácilmente las cosas y no perder la conexión con la representación escrita. Probablemente ha surgido de mi propia costumbre de cambiar las cosas hasta el último momento antes de la actuación y, en muchos casos, incluso durante la presentación. SuperCollider 3 ahora se basa enteramente en un estilo de programación sobre línea de comandos. El sonido está controlado por los mensajes que se envían al servidor y todas las demás construcciones se basan en eso. Podemos esperar que estos *scripts* desarrollen un nuevo estilo de programación en los conciertos. (Collins et al, 2003, p. 328)[27].

Para una explicación ilustrada del uso de JITLIB se puede consultar la ayuda traducida al español por parte del Centro Multimedia del CENART.[28]

La creación en tiempo real dentro de *SuperCollider* es uno de los ámbitos más importantes del programa por lo que no es raro que sigan surgiendo herramientas encaminadas a mejorar las presentaciones en directo. Una de ellas es IxiLang,[29] un ambiente de "super alto nivel" que corre sobre *SuperCollider,* diseñado con la intención de hacer más rápido el proceso de crear sonido. Esta herramienta fue creada por Thor Magnuson, quien también desarrolló una librería para poder usar Impromptu con *SuperCollider*.

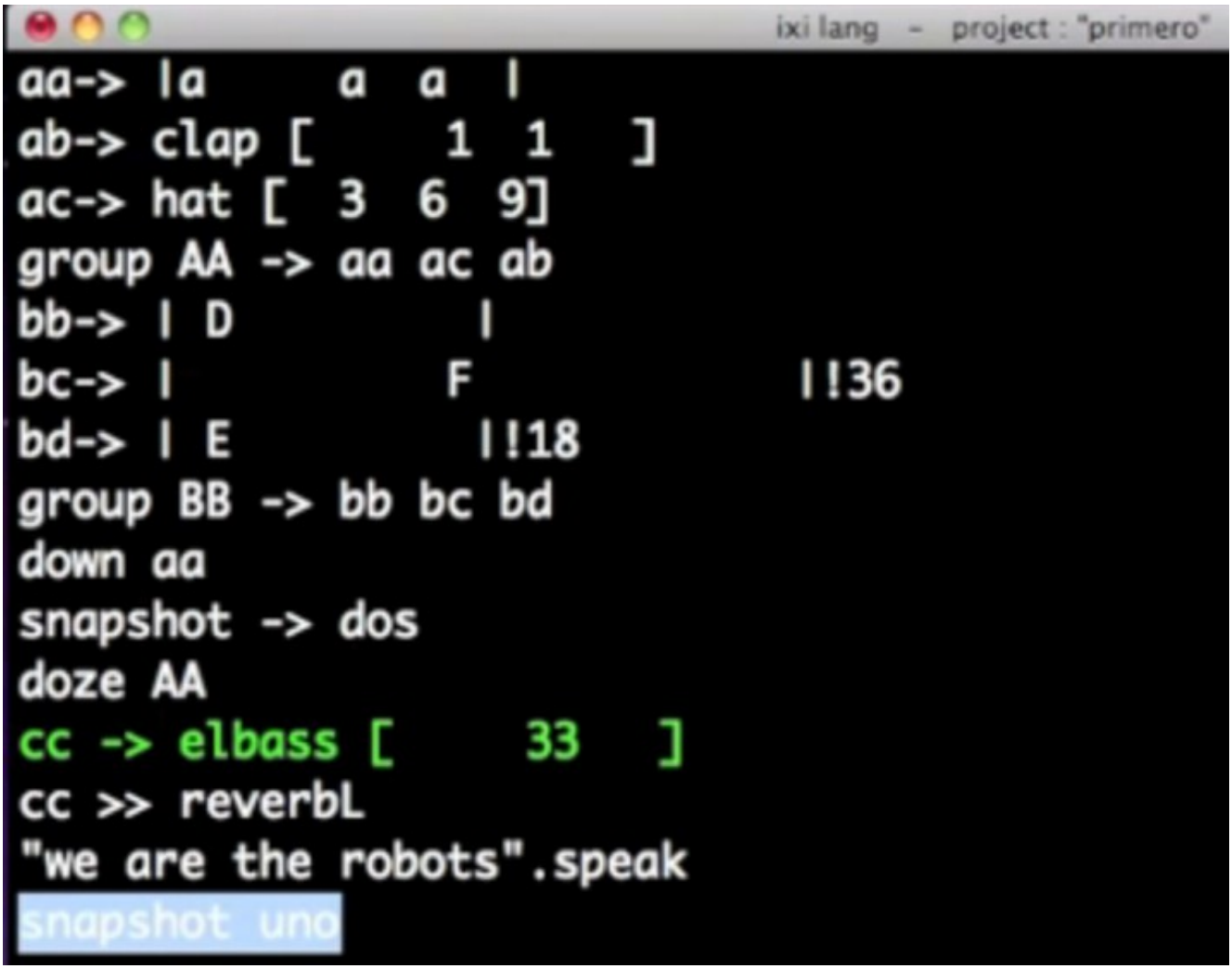
---

[27] Original en inglés: "My reason to introduce the just in time programming library (jitlib) was to make an interface to write code while playing that removes the distinction between preparation and action, so that I could more easily change things and not lose the connection to the written representation. Probably it has emerged from my own habit to change things up until the last moment before the performance and in many cases even during playing. SuperCollider 3 is now based entirely on a style of command-line programming. The sound is controlled by messages that are sent to the sound server and all other constructions are built on that. We can expect these scripts to develop a new style of concert programming."

[28] Referencia: http://cmm.cenart.gob.mx/tallerdeaudio/actividades/sesioneslivecoding/sesioneslivecoding/jitlib.pdf

[29] Referencia: http://www.ixi-audio.net/ixilang/

Vista de IxiLang[30]:



Gracias a herramientas como *JitLib*, *IxiLang*, la posibilidad de utilizar controladores MIDI y muchas prestaciones desarrolladas por la comunidad, *SuperCollider* tiene un gran potencial para el *laptop performance*, y es esta contribución constante a través de diferentes perspectivas lo que le da un futuro promisorio.

3.4 Laptop performance como práctica musical

El uso de computadoras portátiles para una presentación musical es una práctica cada vez

[30] Ejemplo realizado por mí para mostrar el uso de IxiLang el video se puede consultar en: https://vimeo.com/33879670

más común, en el caso de la música no académica en un principio se relacionaba con estilos asociados a la música baile, en la actualidad con la electrónica experimental, síntesis de sonido, e incluso con los errores digitales. “La estética post-digital se encuentra situada entre las formas populares de la música electrónica de baile y las formas modernistas de la academia clásica.” (Turner 2003, p. 81 citado por Zadel,2006)[31]

Ya sea realizando *live coding,* usando controladores externos, utilizando software libre o programas comerciales, el *laptop performance* entra al diálogo con una serie de paradigmas respecto a la creación e interpretación musical. A continuación se presentarán algunos ejemplos.

a) El concepto de habilidad (*skill*) frecuentemente asociado a la interpretación, no es un imponderable en el *laptop performance,* ya que otros conceptos poco comunes en la música académica, como la experimentación en tiempo real, tienen mayor relevancia. La música por computadora en vivo, remite entonces, a una de las expresiones musicales más interesantes de la segunda mitad del siglo XX: la improvisación.

> Las estrategias de improvisación o aquellas formas de música “instantánea” tienen su punto de partida en la idea de flujo sonoro, de evento irrepetible, de momento que puede albergar un universo. Las músicas improvisadas sin duda tienen diferentes raíces... De todos modos, la secuencia que nos interesa en este escrito es la que tiene dos grandes orígenes: uno, el relacionado con el free jazz; el otro con las músicas aleatorias e indeterminadas de la música vanguardista posterior a 1950... que marca la improvisación surgida de músicos con formación académica y que surgiera luego de los descubrimientos de la aleatoriedad y el teatro instrumental posteriores a John Cage. Valiéndose de la exploración no convencional de instrumentos, de las –por entonces- nuevas fuentes de sonido electrónico en vivo y las estrategias de la improvisación pautada a modo de composición, con grado variable de libertad. (Varela, 2005, p.44).

El énfasis se pone en la improvisación o el papel de los artistas intérpretes en la realización de la pieza, utilizando a menudo técnicas aleatorias. A diferencia de la ejecución tradicional donde cada pequeña modulación es controlada físicamente por el instrumentista, a veces en forma automática e inconsciente, los instrumentos digitales pueden ser previamente

[31] Original en inglés: “[T]he post-digital aesthetic finds itself situated between the popular forms of electronic dance music and the modernist forms of the classical academy”

modificados sin restricción o abruptamente; el ejecutante no necesariamente tiene que revisar o realizar estos cambios pues pueden ser condicionados y supervisados por la computadora, puede haber por ejemplo automatizaciones condicionadas a osciladores y/o a números complejos.

La música por computadora en vivo a través de lenguajes de programación y su uso en tiempo real acompaña la búsqueda musical de un espíritu de experimentación científica al seguir una metodología propia de la sintaxis y los algoritmos. Al escribir código, desde el punto de vista creativo, estamos definiendo, describiendo y puliendo nuestro objeto. Después, al ejecutar ese código y mostrarlo estamos definiendo un momento, dando valor a la actividad y no tanto al producto acabado. Desde esa perspectiva, el arte digital, a través del *laptop performance*, se acerca más a un arte-momento que a un arte-objeto, cuestionando la tradición y el paradigma del objeto como obra de arte.

> La capacidad de improvisar tanto micro como macro abstracciones y manipular las combinaciones de procesos de cualquier tipo, proporciona a los programadores en vivo, la oportunidad de explorar nuevas vías de colaboración musical e improvisación. El público recibe no sólo nuevas experiencias sonoras, puede acercarse también a un nuevo nivel de implicación intelectual, posible gracias a la representación explícita de las ideas descritas en el código fuente. (Sorensen, 2005, p. 5).

b) El advenimiento de la computadora digital establece el medio ideal para la escritura de música en forma de algoritmos. No sólo es la música codificada, también es una herramienta de composición en este sentido. Los algoritmos que se derivan pueden entenderse con la misma importancia que John Cage y Xenakis, por ejemplo, le dan a la partitura; una sensación también de obra de arte por sí mismo.

El código está en constante cambio y a menudo se modifica. Por esta razón McLean et al (2011) y Magnusson (2011) hablan de "codeomorphology", en referencia a que el código y la música evolucionan juntos en un proceso que es observado por el público. El algoritmo suele asociarse como una nueva forma de la partitura musical en especial de la gráfica.

> Las partituras gráficas pueden representar una forma especial de algoritmo. Son instrucciones en las que el compositor ha encontrado una razón para ir más allá de la notación tradicional. Muchos compositores que exploraron la obra abierta, como Christian Wolff, Karlheinz

> Stockhausen, John Cage, Cornelius Cardew y Iannis Xenakis, han decidido utilizar la partitura gráfica para ampliar el lenguaje musical. Esto se hace a menudo con el fin habilitar lo no lineal, el aumento en la interpretación del intérprete y presentar elementos sorpresa. La partitura gráfica abre nuevas dimensiones en la ontología de la música al rechazar lo lineal en la notación musical, abriendo así el camino para el código generativo y la música algorítmica. (Magnusson, 2011, p. 21)[32].

En un principio se puede entender a la computadora como un instrumento para la interpretación perfecta, dado que en la partitura se da una descripción de la música mediante diversas anotaciones que representan parámetros musicales como son: tono, velocidad y duración. En este sentido, el código de programación que se deriva del *laptop performance* puede verse como una partitura, entendiendo a la misma como el mensaje del compositor al instrumentista que interpreta la información.

En la cultura occidental se encuentra un fuerte deseo de representar la música, se posee una tradición que busca la codificación y decodificación de datos musicales, y así, almacenar la forma de interpretar la música; es posible que por eso se piense en el algoritmo como una partitura y se busque asimilar al *laptop performance* con las prácticas tradicionales. Por supuesto, prácticas como el *live coding* ofrecen un contexto ideal para acercar a la audiencia a la representación de la pieza, pues al presentar en tiempo real las modificaciones que se realizan al algoritmo, éste se convierte en un elemento artístico y se integra a la propuesta estética de la pieza.

> A mediados del siglo 20 se observaron muchos experimentos intermedia que involucraron medios visuales y de sonido. Artistas como Kandinsky y Klee experimentaron cómo se podría representar un medio sincrónico como la pintura como un proceso diacrónico (como la música). Estos experimentos continuaron con el movimiento Fluxus de los años 60 y la introducción de un nuevo enfoque, a saber, la celebración del algoritmo como una forma de arte. (Magnusson, 2011, p. 20)[33].

[32] Original en inglés: Graphic scores can represent a special form of algorithm. They are instructions with which the composer has found a reason to go beyond traditional notation. Many composers exploring the open work, such as Christian Wolff, Karlheinz Stockhausen, John Cage, Cornelius Cardew and Iannis Xenakis, have resolved to use the graphic score to extend the musical language . This is often done with the aim of enabling non-linearity, increasing performer interpretation and presenting elements of surprise. The graphic score opens new dimensions in the ontology of music by rejecting linearity in musical notation, thus paving the way for encoded generative or algorithmic music.

[33] Original en inglés: The mid-20th century saw many interme- dia experiments involving visual media and sound. Artists such as Kandinsky and Klee experimented with how a syn- chronic medium such as painting could be represented as a diachronic process (such as music). These experiments continued with the Fluxus movement of the 1960s introducing a novel approach, namely the celebration of the algorithm as an art form.

Estas prácticas dentro del *laptop performance* que utilizan lenguajes de programación, y por tanto algoritmos para su ejecución, plantean, quizás, un cambio "De las partituras estáticas y formatos de almacenamiento deterministas, [pasando] a las partituras que se escriben o ejecutan en tiempo real, ocasionando un nuevo tipo de música en vivo: la música generativa." (Magnusson, 2011, p. 23)[34]

En años recientes la capacidad de las computadores personales ha incrementado hasta el punto de hacer posible la creación de música en tiempo real, agilizando los procesos y economizando diferentes aplicaciones.

c)La utilización de lenguajes de programación en tiempo real para crear música ofrece un punto de partida interesante y novedoso respecto a la interpretación musical tradicional. La música por computadora en vivo puede funcionar como un elemento peculiar dentro de la relación composición-ejecución al eliminar esta dicotomía. Al momento que el artista crea una pieza y la ejecuta el ordenador, existe un instante en el que la instrucción pasa por un proceso lógico dentro de la computadora, creando un puente interesante, por un instante la experiencia puede asemejarse a la del compositor que recibe del intérprete lo que el compositor ha (programado) escrito en su partitura, sin embargo, en esa experiencia, la libertad del intérprete es subsecuente a la del compositor, hay dos sujetos. Sin embargo, en la música por computadora, el músico: intérprete y compositor, se convierten en uno sólo. Esto sucede ya que la capacidad de modificar la computadora, ya sea mediante código o mediante algún controlador, es de la misma persona, por lo que el *laptop performance* ofrece la virtual desaparición de la frontera entre el compositor y el intérprete. Dado que la brecha entre la composición e interpretación se vuelve vaga, es evidente que los compositores que utilizan estas técnicas participan de forma activa en la ejecución de su trabajo.

d) Una rápida mirada a los medios de transmisión de esta práctica y a las herramientas que utiliza, dialoga directamente con otro paradigma de la música de arte: la creación individual de una obra musical. El producto del *laptop performance,* a través de prácticas como los conciertos telemáticos (es decir colaboraciones a distancia por medio de Internet), puede derivar en una creación anónima y/o colectiva

[34] Original en ingés: From static scores and deterministic storage formats, we move to scores that are written or executed in real time, resulting in a new type of live music: generative music.

que evita la exaltación de la obra de arte como objeto y la consecuente idea de propiedad (intelectual y material) de la pieza. Quizás para algunas personas la admiración por la obra de arte finalizada y la habilidad del autor no tienen el mismo peso que el interés por la praxis en sí misma.

Un último punto que merece mencionarse es el hecho de que la democratización de este tipo de tecnologías difumina la línea divisoria entre la música académica y la música popular, al acercar las herramientas con todas sus posibilidades a distintos interesados en la creación musical. Esto logra que estas nuevas prácticas sean factibles de considerarse no necesariamente como una ruptura total con la tradición, pero sí como una alternativa y una contribución a la práctica musical. “La historia de toda forma artística pasa por tiempos críticos en los que tiende a urgir efectos que se darían sin esfuerzo alguno en un tenor técnico modificado, esto es, en una forma artística nueva.” (Benjamin,1973, p. 243)

Capítulo IV

Trabajo práctico: BEAT, herramienta para laptop performance en SuperCollider

La computadora es un instrumento que permite crear música y a su vez es un instrumento para la ejecución musical; para estos fines, se utilizan diversos tipos de programas, dentro de los cuales se encuentran los llamados secuenciadores. Duignan, Noble y Biddle (2005) describen una taxonomía de los tipos de interfaces para los usuarios que manejan estos programas, agrupándolos de la siguiente forma:

El primer grupo se refiere a las primeras aplicaciones que eran extensiones y desarrollos construidos en lenguajes de programación como la familia de *MusicN*, *SuperCollider* y extensiones como *Siren* en *SmallTalk*; a éstas se les conoce como herramientas musicales mediante lenguajes de texto.

El segundo grupo abarca programas basados en cajas de ritmo y secuenciadores análogos, dando origen a reproductores de *loops* y de *samples,* este grupo incluye las aplicaciones que usan secuencias por patrones como pueden ser *Reason*, *Ableton Live* o *Hydrogen*.

Existe también el grupo que de forma gráfica realiza estos procesos mediante herramientas de programación visual como *MAX MSP*, *PD* y *VVVV*, los cuales tienen una gran variedad de aplicaciones en el ámbito musical.

Finalmente, la cuarta división es la que se refiere a los secuenciadores lineales que utilizan la metáfora de la grabación multipista, representada en la interfaz gráfica con una línea de tiempo donde se trabaja con los objetos sonoros susceptibles de modificarse o procesarse. Ejemplos claros de estos últimos son *Pro Tools*, *Logic* o *Cubase*. La mayoría de los programas que se han creado bajo este paradigma han sido privativos aunque existen opciones libres como *Ardour*, *LNX Studio*, *Macaw* o *Qtracto* y *Rosegarden*, siendo estos últimos exclusivos para Linux.

Los ejes que se proponen para elaborar esta taxonomía son: medio, nivel de abstracción, etapa de linealización, orden de evento, aplicabilidad, como lo muestra la siguiente tabla (Duignan et al,2005, p. 3) :

Característica		
Medio	Textual	Gráfico
Nivel de abstracción	Predeterminado	Personalizado
Linealización del Evento	Temprana	Retrasada
Ordenación de Eventos	Control	Datos
Aplicabilidad	Específica	General

Las posibilidades que ofrece cada una de estas cinco características permite definir el tipo de interfaz que tiene un secuenciador según la taxonomía que proponen Duignan, Noble y Biddle. (2005)

Por supuesto que los programas combinan características de las diferentes clasificaciones, algunos programas originalmente lineales como Logic o Cubase han incorporado otro tipo de prestaciones. Por ejemplo, en el contexto de una presentación, ciertos elementos de la música se dejan para ser ejecutados en vivo, por lo que estos programas han añadido vistas para este tipo de prácticas. Uno de los programas más usados para la ejecución en tiempo real es *Ableton Live*, dentro de su interfaz incorpora dos vistas, una como secuenciador tradicional y otra que permite lanzar clips de audio o MIDI facilitando una sesión en directo. Es pertinente mencionar también la aplicación *Max for Live* que permite al usuario crear y editar sus propios dispositivos, los cuales son completamente compatibles con la interfaz y el sistema de Live.

Este tipo de herramientas tienen diseños fijos por lo que programas con interfaces adaptables a través de lenguajes de programación surgen también como alternativas que pueden sustituir o complementar la forma en que se utiliza el software de organización lineal. En este sentido, *SuperCollider* aparece como una opción que además permite pensar en el diseño de nuevas herramientas para la música por computadora.

El ordenador y otros dispositivos digitales afectan los procesos de composición, almacenamiento y distribución musical. Estas nuevas tecnologías propician a su vez la búsqueda de nuevas interfaces en el diseño de instrumentos, tendencia que ha encontrado en la informática un campo fértil para la experimentación de nuevas ideas.

A manera de reflexión me remito a lo que John Cage, mencionaba en su texto *Credo*: “La mayoría de los inventores de instrumentos musicales eléctricos intentaron imitar a los instrumentos de los siglos XVIII y XIX, de la misma manera que los diseñadores de

automóviles emularon los carruajes"(Cage, 1958). Y en parte, esto es lo que ha sucedido con los instrumentos virtuales que se han creado para usarse en las computadoras, sin embargo, modificar este patrón de imitar el pasado en el diseño de instrumentos podría generar nuevas formas de composición y desarrollar nuevos elementos en la construcción musical. Por supuesto, el romper estos moldes de una forma que permita generar nuevos paradigmas es un proceso que llevará un largo recorrido.

La parte práctica sobre la cual se apoya el presente trabajo de investigación incluye el diseño de una herramienta creada con *SuperCollider* que permita generar ritmos en primera instancia o secuencias para cualquier sonido creado con el mismo programa. Es importante aclarar que no se pretende mostrar esta herramienta como un rompimiento total con el diseño de herramientas de funciones similares en otros programas, sobre todo en lo que a la parte gráfica se refiere, pero contiene peculiaridades en su núcleo que la diferencian de otras herramientas afines ya que permite utilizar procesos de secuenciación completamente propios de *SuperCollider*; esto se vuelve importante en un nivel del funcionamiento lógico.

En los últimos diez años, con la expansión de la computadora portátil, el *laptop performance* se ha consolidado como una práctica común; se ha diseñado mucho software, y a nivel de hardware, el uso del protocolo MIDI ha otorgado interactividad y flexibilidad. El uso de software en vivo y el uso de controladores son comunes en el *laptop performance,* esta proliferación de presentaciones ha provocado también expresiones de música por computadora fuera del ámbito académico, tendencia que beneficia al campo de la llamada *computer music*, ya que permite mayor desarrollo e interés incluso de tipo comercial en sus herramientas. Esta práctica deriva en que cada vez una mayor cantidad de personas se acerquen a las mismas, por tanto, es conveniente diseñar aplicaciones que coincidan con esta perspectiva.

Dentro de los nuevos alcances de la música por computadora, expresiones como el *live coding,* suman para enriquecer una propuesta, que al utilizar el acto de programar, busca aprovechar el potencial del instrumento (la computadora) en todos sus niveles. Esta perspectiva es uno de los fundamentos del apartado práctico de este trabajo; se busca aprovechar el potencial de *SuperCollider* para diseñar y modificar, desde cualquier nivel, una interfaz personalizada, así como permitir la integración de dicha interfaz con otras herramientas que el creador musical utiliza para el *laptop performance,* ya sea dentro de

SuperCollider o en conjunto con otros programas.

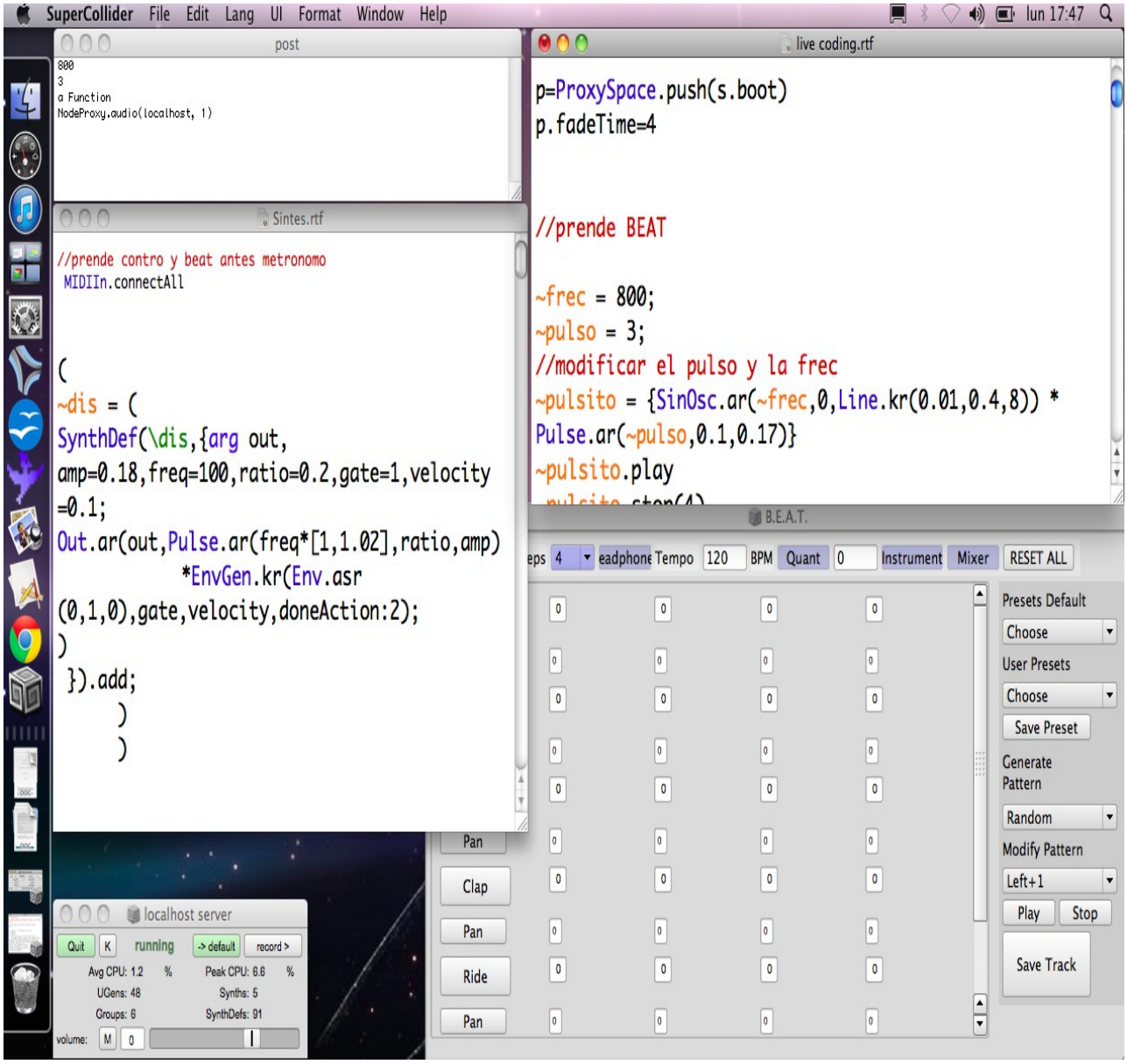
4.1 BEAT (Beat Editor Arrays -Tool- on Time)

El *live coding* por sí mismo ofrece grandes posibilidades para el *laptop performance.* Sin embargo, debido a las necesidades de ciertas presentaciones en vivo, ésta puede no ser la única opción de uso que se le dé a *SuperCollider* como herramienta en directo, principalmente por el peligro que representa correr código complicado en vivo.

En la práctica es común tener contenido preparado, código probado y previamente hecho, esto es similar a lo que pretende la herramienta *BEAT* que es la parte práctica del trabajo, una manera eficiente de definir los patrones para ciertos sonidos en un entorno de actuación. El objetivo es permitir que un músico pueda generar estructuras sobre la marcha en el escenario o con una programación previa y utilizarlas en el momento que prefiera. Es decir, se depende de una carga fuerte de improvisación pero también se permite utilizar *presets* que le hayan parecido interesantes al usuario en algún momento.

Como un ejemplo, para mi proyecto *Bot Cocktail*, *laptop performance* mediante *SuperCollider*, utilizo las prestaciones MIDI de *SuperCollider*, así como modificaciones mediante *live coding* y código previamente preparado así como la herramienta personalizada BEAT.

Captura de pantalla, proyecto Bot Cocktail:

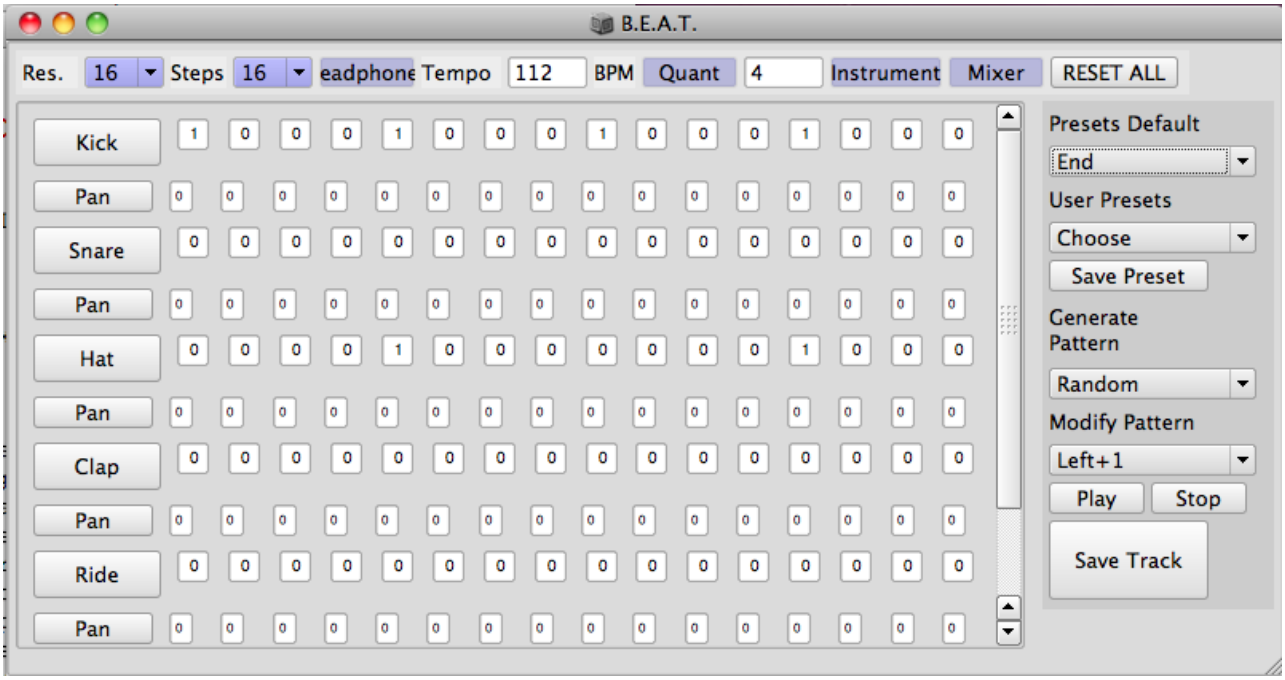


En una presentación de este tipo, ciertos aspectos funcionales se encuentran ya ordenados. En la parte superior derecha se encuentran comandos para activar ciertos sonidos o desaparecerlos, pero a su vez se permiten modificaciones en el momento, haciendo uso de la práctica de escribir código en vivo. Para esto se activa un *ProxySpace*, instrucción que crea un ambiente dentro de *SuperCollider* en el que se pueden declarar variables susceptibles de modificación en cualquier momento mientras este ambiente esté activo.

La parte superior izquierda de la pantalla muestra los mensajes que ofrece el

intérprete de *SuperCollider* y debajo se encuentra el *script* de un instrumento listo para utilizarse con un controlador MIDI; la parte inferior izquierda muestra el estado del servidor de *SuperCollider*; y finalmente, la parte inferior derecha de la pantalla muestra la herramienta *BEAT*.

BEAT, cuyo nombre se puede apreciar en el título, proviene del acrónimo recursivo en inglés *Beat Editor Arrays -Tool- on Time.*



La interfaz permite crear y modificar patrones de sonidos; no resta importancia a la utilización de secuencias de control preprogramados, pero también se mueve hacia la creación de música en el escenario. La facilidad de tener ciertos sonidos o instrumentos de *SuperCollider* previamente diseñados facilita la ejecución de patrones en el programa pero siguiendo la sintaxis básica a través de un *array* (colección) de amplitudes. La idea principal del sistema es que el usuario de *SuperCollider* incorpore la forma de trabajo a través de *arrays* para la creación de secuencias, y por supuesto, pueda utilizar cualquier tipo de instrumento creado por él. En un nivel más bajo de programación, el usuario puede modificar la interfaz y los parámetros que integra.

La interfaz gráfica recupera la esencia de ciertas cajas de ritmo virtuales que han sido diseñadas para otros sistemas operativos, en particular *Hydrogen*, diseñada originalmente

para Linux. Esto tiene la ventaja de ofrecer un acercamiento potencialmente intuitivo al usuario novel de *SuperCollider* o de otros programas, ya que esta representación visual ayuda a establecer una conexión con los usos más tradicionales y conocidos del *laptop performance.* Hay que tomar en cuenta que "los dispositivos utilizados por el artista... no son meramente herramientas inertes, indiferentes a los resultados, que podrían ser reemplazadas por otras. Éstas están cargadas de conceptos, tienen una historia, derivan de condiciones productivas bastante específicas." (Machado, 2006, p. 152) En este sentido, la idea inicial de la herramienta responde a un proceso en el cual, primero y como es natural, se busca satisfacer una necesidad particular, en este caso, para un acto en vivo usando *SuperCollider*.

Para una presentación tal, se toma como base la experiencia en el uso de programas para la producción de música electrónica, respecto al uso de secuencias rítmicas, posteriormente se testearon una buena cantidad de programas con la intención de extraer y entender las características que cumplieran lo ya mencionado, siempre buscando priorizar la amabilidad con el usuario.

BEAT es sólo la primera de un grupo de herramientas, que en conjunto con la programación en tiempo real, pretenden permitir una mejora en satisfacer las necesidades para la ejecución en tiempo real a través de la computadora para los músicos del siglo XXI. Sus parámetros no están diseñados para cumplir con las expectativas de cualquier usuario, sino las necesidades específicas de quien lo está programando; es el núcleo que se puede modificar lo que se convierte en una herramienta cuyo potencial tiene como límite las capacidades y creatividad del usuario, al igual que la programación y el software libre.

BEAT será liberado bajo el uso licencias tipo *GNU GPL*[35] (licencia pública general) y el manual como *GNU FDL*[36] (licencia de libre documentación). La decisión de colocar este código bajo esta licencia en particular se debe a lo importante que resulta mantener una regulación, que prohíba una indebida apropiación de la herramienta y por la posibilidad de heredar la responsabilidad a las personas que decidan utilizar la aplicación y modificar su código fuente. Pero la razón más importante de poner una licencia de este tipo es contribuir a la continuación de estos mecanismos a través internet, pues el desarrollo del software libre

[35] Garantiza a los usuarios finales (personas, organizaciones, compañías) la libertad de usar, estudiar, compartir (copiar) y modificar el software. Licencia completa: http://www.gnu.org/copyleft/gpl.html

[36] Permite la libre copia y distribución del manual pero no así los cambios al mismo. Licencia completa: http://www.gnu.org/copyleft/fdl.html

ha ayudado y acelerado procesos en la informática de formas incuantificables. Además la apertura y los sistemas abiertos son fundamentales para lo que representa una democratización del conocimiento.

4.2 Manual de usuario BEAT

Una cuestión que a veces dificulta el acercamiento al software libre y desarrollado a través de una comunidad es la falta de la documentación adecuada, por lo que a continuación se incluye una descripción de todas y cada una de las funciones gráficas de *BEAT*, en forma de manual de usuario, el cual será distribuido junto con la aplicación en archivo *.pdf* y archivo de ayuda de *SuperCollider*.

BEAT

(Beat Editor Arrays -Tool- on Time)

Carlos Octavio Gutiérrez

BEAT es una herramienta escrita en *SuperCollider*, que permite emular una caja de ritmos, para lo cual utiliza una secuencia de datos (*arrays*) que corren dentro de patrones (*pdefs*). Para su creación se ha utilizado la versión 3.6.5 de *SuperCollider*; por el momento está probado en *MAC OSX*, aunque se invita a los usuarios a probarlo en Windows y Linux para verificar su funcionalidad.

Contenido.

Leer antes de usar.

1. La carpeta completa de archivos puede descargarse del repositorio de *GITHUB*,[37] así como solicitarse por correo a la siguiente dirección de correo electrónico: *vjtavo@gmail.com*
2. Se debe instalar la carpeta completa en la ruta de extensiones de su sistema operativo. Dicha ruta se puede obtener mediante el comando “Platform.systemExtensionDir;”. Antes de instalar, eliminar el sufijo -master del nombre de la carpeta. Una vez instalada la carpeta se debe recompilar la librería, en caso de tener activo *SuperCollider* (“Cmd + K” en Mac).
3. Para usuarios familiarizados con *SuperCollider*, la carpeta “BEAT” contiene una subcarpeta llamada “Files” dónde se encuentran almacenados los archivos que contienen los patrones de cada “Preset”. Estos archivos sólo pueden accederse a través de *SuperCollider*, usándose como diccionarios, con lo cual se pueden borrar *Presets* o mediante la interfaz de

[37] https://github.com/vjtavo/BEAT

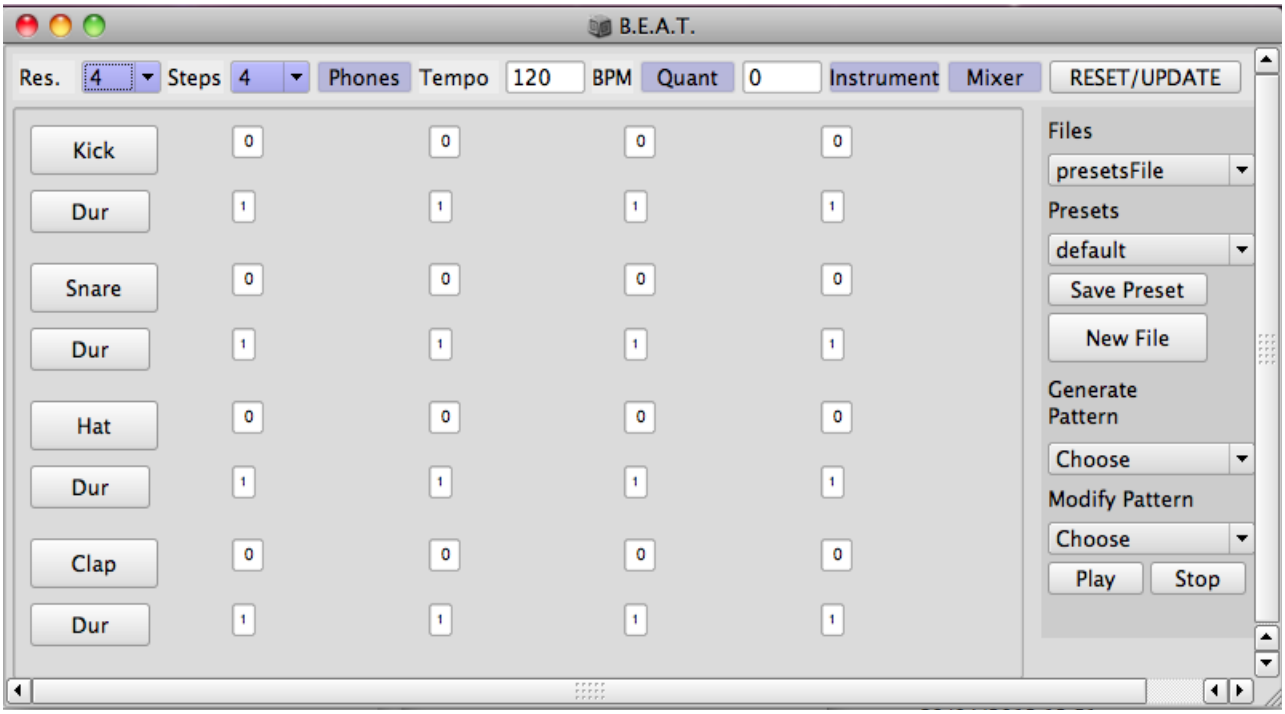
BEAT para almacenar y modificar los mismos.

***IMPORTANTE**: Nunca nombrar un preset: "DATA" o "Data", esto borraría los datos almacenados del resto de presets.*

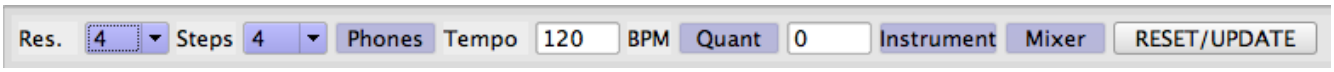
Iniciar BEAT.

Para iniciar, *BEAT* se debe iniciar *SuperCollider* y posteriormente escribir la instrucción "BEAT()" o "BEAT.new" en un documento vacío de *SuperCollider* y evaluarlo.

1. VISTA GENERAL DE BEAT



Barra superior de control



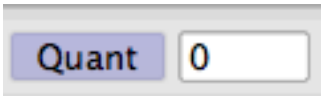
De izquierda a derecha:

La resolución de cada paso del patrón rítmico que se está tocando, va de 4 (cuartos) a 32T (treintaidosavos de tresillo).

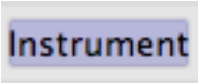
El número de pasos que ejecutará el patrón rítmico.

Activa la salida de audio para audífonos, solo disponible con interfaces de audio compatibles.

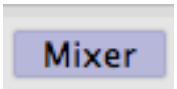
Velocidad de reproducción en Beats por Minuto.



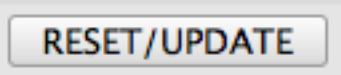
Habilita y recibe un valor para la cuantización



Abre la ventana “Instrument” (Descripción de la ventana en la siguiente sección).



Abre la ventana “Mixer” (Descripción de la ventana en la siguiente sección).



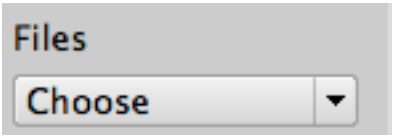
Reinicia “BEAT”, devuelve todo al patrón por default y actualiza la creación de nuevos archivos.

Vista editor de Arrays

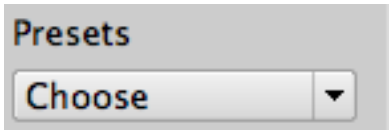
Cuadros de texto en los que se muestran los instrumentos (Synthdefs) que se pueden utilizar, debajo de los cuales otro cuadro texto indica el *array* que modifica la duración de cada paso. Para el usuario familiarizado con *SuperCollider*, un Synthdef puede agregarse, eliminarse o modificarse desde el archivo de clase "BEAT"[38]

[38] Próximamente se agregará un archivo por separado en dónde se almacenen los Synthdef's

Cuadros numéricos en los que se depositan los valores para los distintos *arrays* de cada instrumento, cada cuadro representa un golpe percusivo de un instrumento, la primer línea horizontal recibe los valores de amplitud dentro de un rango de 0.0 a 1, la segunda línea horizontal representa la disposición de paneo del instrumento recibe valores de un rango entre -1 y 1 donde -1 representa canal izquierdo y 1 canal derecho.

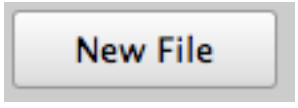


Menú desplegable en el que se selecciona el archivo que se utilizará para buscar Presets.

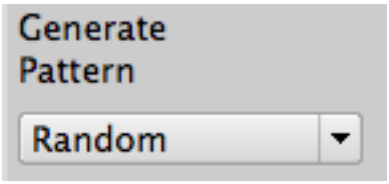


Menú desplegable en el que se selecciona el Preset que se utilizará.

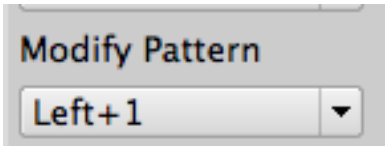
Permite almacenar un patrón rítmico creado con BEAT.



Crea un nuevo archivo para almacenar Presets dentro de la subcarpeta “Files”.



Genera un patrón rítmico, incorpora los principales métodos para patrones en SuperCollider, pero permite al usuario programar generadores dentro de la sección “// Generate pattern” en el archivo de clase “BEAT”.



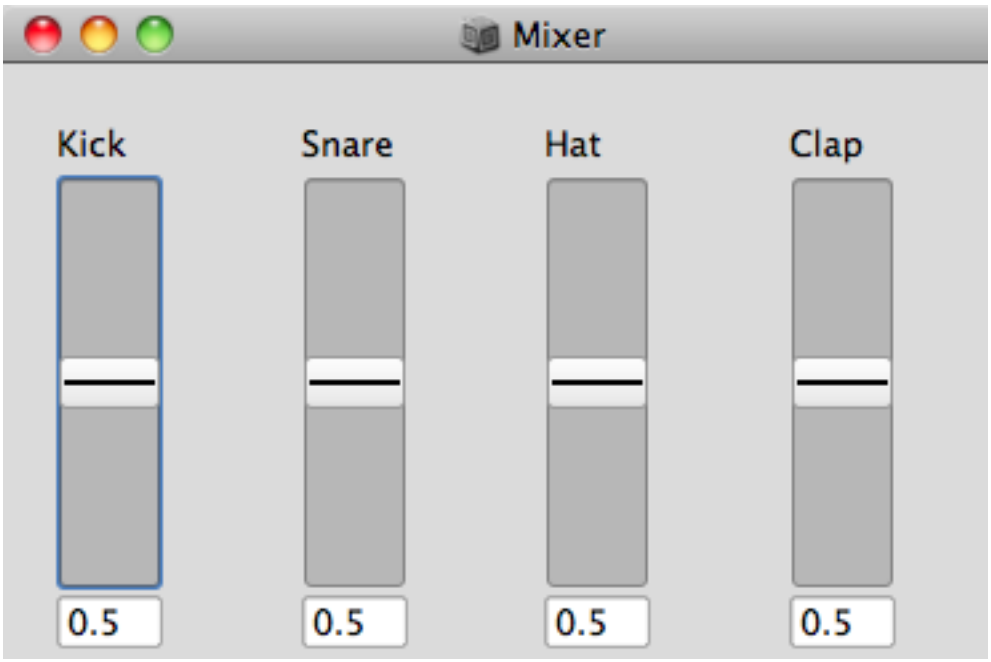
Modifica un patrón rítmico, incorpora una opción para mover el patrón a la derecha otro a la izquierda y uno más para reducirlo a un golpe, pero permite al usuario programar modificadores dentro de la sección “// Modify pattern” en el archivo de clase “BEAT”.

Botones de control del patrón rítmico.

2. VENTANAS AUXILIARES

La ventana *Instrument* permite al oprimir el botón “Synthdef” abrir el archivo en donde se debe agregar el Synthdef del usuario, con las características que se detallan al inicio del

manual, por otro lado el botón “Load File” permite cargar una muestra de audio, Mono o Estéreo, para utilizarse como un instrumento que carga un buffer en *SuperCollider*.



La ventana “Mixer” muestra una serie de sliders que permiten aumentar o disminuir la amplitud de cada instrumento.

En breve se subirá un pequeño video mostrando lo sencillo que es usar *BEAT* para ayudar a un acto en vivo que use *SuperCollider*.

Invitamos a los usuarios a compartir sus archivos de presets en la carpeta *presetsFile* del repositorio oficial: *https://github.com/vjtavo/BEAT.*

Anexo

Código Fuente de BEAT

```
//B.E.A.T.
//(Beat Editor Arrays -Tool- on Time)

BEAT{

var beat, reset, guiInstMix, guiInstBeat, guiMain;

*new {
     ^super.new.initBEAT()
     }
initBEAT{

beat={


var numSteps = 4, maxNumSteps = 64, prueba;
var salida=0;
var vquant=0;
var stretch=1;
var inst1, t1,menuRes, funcs, menuSteps, menuInst1,
      amp, ampNbox, degNbox, octava, inst2, editInst, mixer;
var instruments, update, updateResolution, updateSteps, updateQuant, updateTempo,
updatePresets, tempo;
var header, main, scrollView, makeSteps, options, quantButton, pdefs, bpm, gain;
var res, steps, headphone, nombreDur, instrument,stop, play, presets, currentPreset =
\default;
var bufferInstrumentMaker, pdefMaker, updatePdefs, presetsFile, savePresets, saveFile,
generate;
var nomPreset, nomFile, presFold, arrayFiles, defFile,
modify, menMod;
var filePresets, presetToSave, instrumentsSel, menGen,
saveButton, pathFile, selFile;
var menFiles, updateFileUsed, menPres, arrayPresets,
defPres, dataPreset, instMix, instBeat;


instruments = List[ ];
instrumentsSel = List[ ];


//Synthdef
// by otophilia
SynthDef("Kick", {
          arg outBus=0, amp;
          var env0, env1, env1m, out;

          env0 =  EnvGen.ar(Env.new([0.5, 1, 0.5, 0], [0.005, 0.06, 0.26], [-4, -2, -4]),
doneAction:2);
```

```
            env1 = EnvGen.ar(Env.new([110, 59, 29], [0.005, 0.29], [-4, -5]));
            env1m = env1.midicps;

            out = LFPulse.ar(env1m, 0, 0.5, 1, -0.5);
            out = out + WhiteNoise.ar(1);
            out = LPF.ar(out, env1m*1.5, env0);
            out = out + SinOsc.ar(env1m, 0.5, env0);

            out = out * 1.2 * amp;
            out = out.clip2(1);

            Out.ar(outBus, out.dup);
      }).add;
      instruments.add(\Kick);
//
      SynthDef("Snare", {
            arg outBus=0, amp=0.8;
            var env0, env1, env2, env1m, oscs, noise, out;

            env0 = EnvGen.ar(Env.new([0.5, 1, 0.5, 0], [0.005, 0.03, 0.10], [-4, -2, -4]));
            env1 = EnvGen.ar(Env.new([110, 60, 49], [0.005, 0.1], [-4, -5]));
            env1m = env1.midicps;
            env2 = EnvGen.ar(Env.new([1, 0.4, 0], [0.05, 0.13], [-2, -2]), doneAction:2);

            oscs = LFPulse.ar(env1m, 0, 0.5, 1, -0.5) + LFPulse.ar(env1m * 1.6, 0, 0.5, 0.5,
-0.25);
            oscs = LPF.ar(oscs, env1m*1.2, env0);
            oscs = oscs + SinOsc.ar(env1m, 0.8, env0);

            noise = WhiteNoise.ar(0.2);
            noise = HPF.ar(noise, 200, 2);
            noise = BPF.ar(noise, 6900, 0.6, 3) + noise;
            noise = noise * env2;

            out = oscs + noise;
            out = out.clip2(1) * amp * -3.dbamp;

            Out.ar(outBus, out.dup);
      }).add;
      instruments.add(\Snare);
//
SynthDef("Hat", {
            arg outBus=0, amp=0.3;
            var env1, env2, out, oscs1, noise, n, n2;

            n = 5;
            thisThread.randSeed = 4;

            env1 = EnvGen.ar(Env.new([0, 1.0, 0], [0.001, 0.2], [0, -12]));
            env2 = EnvGen.ar(Env.new([0, 1.0, 0.05, 0], [0.002, 0.05, 0.03], [0, -4, -4]),
doneAction:2);

            oscs1 = Mix.fill(n, {|i|
            SinOsc.ar(
```

```
                    ( i.linlin(0, n-1, 42, 74) + rand2(4.0) ).midicps,
                    SinOsc.ar( (i.linlin(0, n-1, 78, 80) + rand2(4.0) ).midicps, 0.0, 12),
                    1/n
               )
          });

          oscs1 = BHiPass.ar(oscs1, 1000, 2, env1);
          n2 = 8;
          noise = WhiteNoise.ar;
          noise = Mix.fill(n2, {|i|
               var freq;
               freq = (i.linlin(0, n-1, 40, 50) + rand2(4.0) ).midicps.reciprocal;
               CombN.ar(noise, 0.04, freq, 0.1)
          }) * (1/n) + noise;
          noise = BPF.ar(noise, 6000, 0.9, 0.5, noise);
          noise = BLowShelf.ar(noise, 3000, 0.5, -6);
          noise = BHiPass.ar(noise, 1000, 1.5, env2);

          out = noise + oscs1;
          out = out.softclip;
          out = out * amp * -9.dbamp;

          Out.ar(outBus, out.dup);
     }).add;
     instruments.add(\Hat);
//
SynthDef("Clap", {
          arg outBus=0, amp = 0.5;
          var env1, env2, out, noise1, noise2;

          env1 = EnvGen.ar(Env.new([0, 1, 0, 1, 0, 1, 0, 1, 0], [0.001, 0.013, 0, 0.01, 0,
0.01, 0, 0.03], [0, -3, 0, -3, 0, -3, 0, -4]));
          env2 = EnvGen.ar(Env.new([0, 1, 0], [0.02, 0.3], [0, -4]), doneAction:2);

          noise1 = WhiteNoise.ar(env1);
     noise1 = HPF.ar(noise1, 600);
          noise1 = BPF.ar(noise1, 2000, 3);

          noise2 = WhiteNoise.ar(env2);
     noise2 = HPF.ar(noise2, 1000);
     noise2 = BPF.ar(noise2, 1200, 0.7, 0.7);

          out = noise1 + noise2;
     out = out * 2;
          out = out.softclip * amp * -6.dbamp;

     Out.ar(outBus, out.dup);
}).add;
  instruments.add(\Clap);


//SynthDef("yourSynthdef", { arg amp= ... }).add;
//instruments.add("yourSynthdef");
```

```
//SynthDef Buffer

SynthDef("BEATPlayStereo",
                {| bufnum, amp, gate = 1, dur |
                   var signal, env;

            signal        = PlayBuf.ar(2, bufnum, BufRateScale.kr(bufnum));
            env           = EnvGen.ar(
                                        Env.asr(0.01, 0.7, 0.1),
                                        gate,
                                        doneAction: 2
                                ) * amp;

            Out.ar(0, signal * env)
        }
).add;
//
SynthDef("BEATPlayMono",
                {| bufnum, amp, gate = 1, dur |
                   var signal, env;

            signal        = PlayBuf.ar(1, bufnum, BufRateScale.kr(bufnum));
            env           = EnvGen.ar(
                                        Env.asr(0.01, 0.7, 0.1),
                                        gate,
                                        doneAction: 2
                                ) * amp;

            Out.ar(0, Pan2.ar(signal * env, amp))
        }
).add;

// crea archivo default

if(File.exists("/Library/Application Support/SuperCollider/Extensions/BEAT/Presets
User/presetsFile.rtf"), {},{
                                                        filePresets = ();
                                                        presetToSave = ();
                                                        instruments.collect({
                                                             | instru |
                                                        presetToSave.add
                                                (instru.asSymbol ->
                ( 'amps':Array.fill(64, { 0 }),
     'durs':Array.fill(64, { 1 }),                                                  'gain': 1 ));
                                                         });
                                                        filePresets.add(\default
                                                        -> presetToSave);
                                                        filePresets.add(\Data ->
                                                ());
                                                        filePresets[\Data].add
                                            (\defaultData -> ());
                                                        filePresets[\Data]
                                        [\defaultData].putPairs
                    ([ \Dres, 1, \Dsteps, 4,
```

```
          \Dtempo, 120, \Dquant, 0]);
     filePresets.writeArchive
     ("/Library/Application Support/SuperCollider/Extensions/BEAT/Presets
User/presetsFile.rtf");
                                                            });


// Crea valores iniciales
// para crear el popup menu de archivos de presets
presFold = PathName("/Library/Application Support/SuperCollider/Extensions/BEAT/Presets
User");
arrayFiles = List[ ];

presFold.filesDo{|afile|
     arrayFiles.add(afile.fileNameWithoutExtension.asSymbol);
     };

arrayFiles.do({ arg item, ind; if(item==\presetsFile,
                          {defFile = ind;},
                                                            {})
                                                             });
selFile = \presetsFile;
pathFile = "/Library/Application Support/SuperCollider/Extensions/BEAT/Presets User/"+
+selFile.asString++".rtf";
presets=Object.readArchive(pathFile);
arrayPresets = presets.order;
arrayPresets = arrayPresets.takeThese({ arg item; item.value == \Data; });
arrayPresets.do({ arg item, ind; if(item==\default,
                     {defPres = ind;},
                                                            {})
                                                             });



//GUI
guiMain = Window.new("B.E.A.T.", Rect(510, 455, 800, 400), scroll:true).front;
header = CompositeView(guiMain, Rect(4, 4, 815, 28));
header.background = Color.gray(0.9);
header.decorator = FlowLayout(header.bounds);


res = StaticText(header, 35@20)
                .string_( "Res.")
                .background_(Color.new255(237, 237, 237));

menuRes = PopUpMenu(header, 50@20)
        .items_([ "4" , "8" , "16" , "32", "64", "4T","8T","16T","32T"])
                .action_({| view |
                     update.value(\resolution, view.value)})
                .background_(Color.new255(180, 180, 255));

steps = StaticText(header, 35@20)
                .string_("Steps")
                .background_(Color.new255(237, 237, 237));
```

```
menuSteps=PopUpMenu(header,50@20)
        .items_((1..64).collect({| x | x.asString }))
                .action_({| view | update.value(\steps, view.value + 1) })
                .background_(Color.new255(180, 180, 255))
                .value_(numSteps - 1);


headphone = Button(header,  60@20)
        .states_([
            ["Phones", Color.black, Color.blue(0.7,0.2)],
            ["Phones", Color.black, Color.blue(0.7,0.8)]
            ])
            .action_({|state|
                                    if(state.value == 1,{salida = 2},
                                        {salida = [0,1]})
                                                  });
StaticText(header, 50@20)
                .string_("Tempo")
                .background_(Color.new255(237, 237, 237));


bpm =NumberBox(header, 50@20).action_({| view | update.value
                      (\tempo, view.value/60)})
                                                    .value_(120);


StaticText(header, 26@20)
                .string_("BPM")
                .background_(Color.new255(237, 237, 237));


 Button(header,  60@20)
        .states_([
            ["Quant", Color.black, Color.blue(0.7,0.2)],
            ["Quant", Color.black, Color.blue(0.7,0.8)]
            ]);

quantButton =NumberBox(header, 50@20).value_(vquant.value)
                                            .action_({| view | update.value
                                  (\quantization, view.value)
                                            ;
                       });

editInst = Button(header, 70@20)
        .states_([
            ["Instrument", Color.black, Color.blue(0.7,0.2)],
            ["Instrument", Color.black, Color.blue(0.7,0.8)]
            ])
            .action_({ |val| if(val.value==1,
                instrument
                     )});

mixer = Button(header,  60@20)
```

```
        .states_([
            ["Mixer", Color.black, Color.blue(0.7,0.2)],
            ["Mixer", Color.black, Color.blue(0.7,0.8)]
            ])
            .action_({|val| if(val.value==1,
                    mixer
                                    )});

Button(header, 120@20).states_([[ "RESET/UPDATE" ]])
                                    .action_({reset.value});

scrollView = ScrollView(guiMain, bounds: Rect(4, 36, 632, 345));


makeSteps = { var scrollViewView;

                            if (scrollView.canvas.notNil)
     { scrollViewView.remove };
                            scrollViewView = View();
                            scrollViewView.layout = VLayout();
                            instruments.do{| nombre |
                                 var horizontal = HLayout();
                                 var horizontalDos = HLayout();
                                 horizontal.add(
                                      Button.new.minWidth_(80)
                                                      .minHeight_(30)
                                                      .states_([[ nombre ],[ nombre,
                                                Color.white, Color.blue ]])
                                                      .action_({|state|
                                                                if(state.value == 1,{
                                                               instrumentsSel.add
                                                (nombre.asSymbol);
                                                                },{
                                                                instrumentsSel.remove
                                                      (nombre.asSymbol);
                                                                })
                                                      }),
                                      0,
                                      \topLeft
                                      );
                                      horizontalDos.add(
                                      Button.new.minWidth_(40).minHeight_
                 (10).states_([[ "Dur" ]]),
                                      0,
                                      \topLeft;
                                      );

                                 numSteps.do{| indice |
                                                horizontal.add(
                                   NumberBox.new.minWidth_(20).maxWidth_(20)
                                      .action_({| numberBox |
                                                      presets[currentPreset]
                                      [nombre][\amps][indice] =
                                  numberBox.value;
```

```
                                                        })
                                                .value_(presets[currentPreset]
          [nombre][\amps][indice]).clipLo_(0).clipHi_(1)
                                                 .align_(\right).font_(Font
                                       ("Helvetica", 10)), 0,\topLeft
                                                );

horizontalDos.add(
                                          NumberBox.new.minWidth_(10).maxWidth_
                           (15)
                                          .value_(presets[currentPreset]
                                [nombre][\durs][indice])
                                          .clipLo_(0).clipHi_(4)
                                          .scroll_step_(0.1)
                                          .action_({|numberBox|
                                               presets[currentPreset][nombre]
                                [\durs][indice] = numberBox.value;
                                          })
                                          .align_(\right)
                                          .font_(Font     ("Helvetica", 8)),
                                    0,
                                          \topLeft;
                                          );

                                 };
                                 scrollViewView.layout.add(horizontal);
                                 scrollViewView.layout.add(horizontalDos);
                                 scrollViewView.layout.add(nil);

                           };
          scrollView.canvas = scrollViewView;

};

options = CompositeView(guiMain, Rect(647, 36, 145, 320));
options.background = Color.gray(0.8);
options.decorator = FlowLayout(options.bounds);

StaticText(options, 100@20).string_("Files")
                                                    .align_(\centered);

menFiles = PopUpMenu(options,130@20)
                                 .items_(arrayFiles.collect({|items|
                      items}))
                                 .value_(defFile)
                                 .action_({| menu |
                                 update.value(\fileused,
                                      menu.item)});

StaticText(options, 100@20).string_("Presets")
                                                    .align_(\centered);


menPres = PopUpMenu(options,130@20)
```

```
                                        .items_(arrayPresets.collect({
                                             |items| items}))
                                        .value_(defPres)
                                        .action_({| view |
                                                                  update.value(\presets,
                                                             view.item)
                                             });


Button(options, 100@20).states_([[ "Save Preset" ]])
                                             .action_({ savePresets.value});

Button(options, 100@30).states_([[ "New File" ]])
                                             .action_({saveFile.value});
StaticText(options, 100@40).string_("Generate Pattern");

menGen = PopUpMenu(options,130@20).items_(["Choose", "Random", "1,0,1,0",      "All to 1"])
                                                .action_({ | view |
                                                  generate.value
                                    (view.value)});

StaticText(options, 100@20).string_("Modify Pattern")
                                                  .align_(\centered);

menMod = PopUpMenu(options,130@20).items_(["Choose", "Left+1", "Right+1", "All to 0"])
                                                  .action_({ | view |
                                                 modify.value(view.value)});


Button(options, 60@20).states_([[ "Play" ]]).action_{play.value};

Button(options, 60@20).states_([[ "Stop" ]])
                                             .action_({stop.value});

options.decorator.nextLine;


//Presets


pdefs= List[];
presets = Object.readArchive(pathFile);

pdefMaker = {| instrAsoc |   instrAsoc.key;
                              if (instrAsoc.value[\buffer].isNil)
     {
     pdefs.add(
     Pdef(instrAsoc.key,
          Pbind(
               \instrument, instrAsoc.key,
               \amp, Pn(Plazy({ Pser(instrAsoc.value[\amps]*instrAsoc.value[\gain],
numSteps) }), inf),
               \out, salida,
               \stretch, Pfunc{ stretch },
```

```
                \dur,1 * Pn(Plazy({ Pser(instrAsoc.value[\durs], numSteps) }), inf)
            )
    ).play(quant: vquant.value);

    )
     }

     {
     pdefs.add(
     Pdef(instrAsoc.key,
          Pbind(
               \instrument, instrAsoc.value[\synthdef],          \bufnum,
instrAsoc.value[\buffer].bufnum,
               \amp, Pn(Plazy({ Pser(instrAsoc.value[\amps]*instrAsoc.value[\gain],
numSteps) }), inf),
               \out, salida,
               \stretch, Pfunc{ stretch },
               \dur,1 * Pn(Plazy({ Pser(instrAsoc.value[\durs], numSteps) }), inf)
          )
    ).play(quant: vquant.value);
   )
     };
};
instruments.do{| nombre, ind |



pdefMaker.value(nombre -> presets[currentPreset][nombre]);

};

// Funciones Save

saveFile = {
Dialog.savePanel({ arg path;
                              filePresets = ();
                              presetToSave = ();
                              instruments.collect({| instru |
                              presetToSave.add     (instru.asSymbol ->
          ( 'amps':Array.fill(64, { 0 }),
     'durs':Array.fill(64, { 1 }),                                          'gain': 1 ));
                               });
                              filePresets.add(\default -> presetToSave);
                              filePresets.add(\Data -> ());
                              filePresets[\Data].add(\defaultData -> ());
                              filePresets[\Data][\defaultData].putPairs
                              ([ \Dres, 1, \Dsteps, 4,
                    \Dtempo, 120, \Dquant, 0]);
               filePresets.writeArchive
(path++".rtf");

},{
    "cancelled".postln;
});
```

```
};


savePresets={
var ventPreset, viewPreset, dataToSave;


presetToSave = presets[currentPreset];


ventPreset = Window.new("Save Preset",Rect(100,Window.screenBounds.height-400,
280,100)).front;

viewPreset = TextView(ventPreset.asView,Rect(10,10, 270,30));

Button(ventPreset, Rect(10, 70, 100, 20)).states_([[ "Save" ]])
.action_{saveButton.value;};

Button(ventPreset, Rect(110, 70, 100, 20)).states_([[ "Cancel" ]])
                                        .action_{"Cancelled".postln;
                                             ventPreset.close};

saveButton = {
                   nomPreset = viewPreset.string;
                          dataToSave = nomPreset++"Data";
                          presets = Object.readArchive(pathFile);
                          presets.atFail(\Data,
                           {presets.add(\Data -> ())});
            presets.add(nomPreset.asSymbol -> presetToSave);
              presets[\Data].add(dataToSave.asSymbol -> ());
                     presets[\Data][dataToSave.asSymbol].putPairs                    ([
\Dres, menuRes.value,\Dsteps,numSteps.value,
     \Dtempo,bpm.value,\Dquant, vquant.value]);
                          presets.writeArchive(pathFile);
                          ventPreset.close;

                          };
};


// Init
makeSteps.value;


//Updates
play = { pdefs.do{| pdef | pdef.resume }};
stop = { pdefs.do{| pdef | pdef.pause }};

generate = {| view |
                     var arrayTemp, sizeArray;
                     switch(view,
                                        0, {  },

                                        1, {instrumentsSel.do({| nombre, ind|
```

```
                                        arrayTemp = presets[currentPreset]
                                    [nombre][\amps];
                                         sizeArray = arrayTemp.size;
                                        arrayTemp = Array.fill(sizeArray,
                                { rrand(0, 1)});
                                         presets[currentPreset][nombre]
                                [\amps] = arrayTemp;
                                         makeSteps.value;
                                        });
                                        instrumentsSel = List [];
                                        menGen.value_(0);

                                        },

                                          2,{instrumentsSel.do({|nombre, ind|
                                        arrayTemp = presets[currentPreset]
                                    [nombre][\amps];
                                         sizeArray = arrayTemp.size;
                                        arrayTemp = Array.fill(sizeArray,
                                Pseq([1, 0], inf).iter);
                                         presets[currentPreset][nombre]
                                [\amps] = arrayTemp;
                                         makeSteps.value;
                                        });
                                        instrumentsSel = List [];
                                        menGen.value_(0);

                                        }    ,

                                          3,{instrumentsSel.do({|nombre, ind|
                                        arrayTemp = presets[currentPreset]
                                    [nombre][\amps];
                                         sizeArray = arrayTemp.size;
                                        arrayTemp = Array.fill(sizeArray,
                                { 1 });
                                         presets[currentPreset][nombre]
                                [\amps] = arrayTemp;
                                         makeSteps.value;
                                        });
                                        instrumentsSel = List [];
                                        menGen.value_(0);

                                        }
                                )
                        };

modify = {| view |
                        var arrayTemp;
                        switch(view,
                                        0, {  },

                                        1, {instrumentsSel.do({| nombre, ind|
                                        arrayTemp = presets[currentPreset]
                                    [nombre][\amps];
```

```
                                        arrayTemp = arrayTemp.rotate(-1);
                                         presets[currentPreset][nombre]
                                    [\amps] = arrayTemp;
                                         makeSteps.value;
                                        });
                                        instrumentsSel = List [];
                                        menMod.value_(0);

                                        },

                                    2, {instrumentsSel.do({| nombre, ind|
                                        arrayTemp = presets[currentPreset]
                                    [nombre][\amps];
                                        arrayTemp = arrayTemp.rotate(1);
                                         presets[currentPreset][nombre]
                                    [\amps] = arrayTemp;
                                         makeSteps.value;
                                        });
                                        instrumentsSel = List [];
                                        menMod.value_(0);

                                        },

                                    3, {instrumentsSel.do({|nombre, ind|
                                        var sizeArrayMod;
                                        arrayTemp = presets[currentPreset]
                                    [nombre][\amps];
                                         sizeArrayMod = arrayTemp.size;
                                        arrayTemp = Array.fill
                                        (sizeArrayMod ,{ 0 });
                                         presets[currentPreset][nombre]
                                    [\amps] = arrayTemp;
                                         makeSteps.value;
                                        });
                                        instrumentsSel = List [];
                                        menMod.value_(0);

                                        }

                                    )
                            };


update  = {| type, argument |

                            switch(type,
                                 \resolution, { updateResolution.value(argument) },
                                 \steps, { updateSteps.value(argument) },
                                 \tempo, {updateTempo.value(argument) },
                                 \quantization, {updateQuant.value(argument) },
                                 \presets, {updatePresets.value(argument) },
                                 \fileused, {updateFileUsed.value(argument) }

                            );
```

```
};

updateResolution = {| view |
                                    stretch = switch(view.value,
                                                    0, { 1 },
                                                    1, { 0.5 },
                                                    2, { 0.25 },
                                                    3, { 0.125 },
                                                    4, { 0.0625 },
                                                    5, { 4/3 },
                                                    6, { 0.5 * (4/3) },
                                                    7, { 0.25 * (4/3) },
                                                    8, { 0.125 * (4/3) }
                                                );
};

updateSteps = {| value |
                            numSteps = value;
                            makeSteps.value;
};




updateQuant = {| view |
                            vquant = view.value;
                        pdefs.do{| pdef | pdef.quant_(vquant);
                          };
};

updateTempo = {| value |
                            tempo = value.value;
                            TempoClock.default.tempo = tempo;
};


updatePresets = {| argument |
                                                    dataPreset =
                                    argument.value.asString++"Data";
                                                    numSteps = presets[\Data]
                                              [dataPreset.asSymbol][\Dsteps];
                                                   menuSteps.value = numSteps - 1;
                                         menuRes.value = presets[\Data]
                                    [dataPreset.asSymbol][\Dres];
                                                 stretch = switch(menuRes.value,
                                                    0, { 1 },
                                                    1, { 0.5 },
                                                    2, { 0.25 },
                                                    3, { 0.125 },
                                                    4, { 0.0625 },
                                                    5, { 4/3 },
                                                    6, { 0.5 * (4/3) },
                                                    7, { 0.25 * (4/3) },
                                                    8, { 0.125 * (4/3) }
```

```
                                                    );
                                                            currentPreset = argument;
                                                             vquant = presets[\Data]
                                                [dataPreset.asSymbol][\Dquant];
                                                         quantButton.value = vquant;
                                                             bpm.value= presets[\Data]
                              [dataPreset.asSymbol][\Dtempo];
                                                      TempoClock.tempo = presets[\Data]
                         [dataPreset.asSymbol][\Dtempo]/60;
                                                             updatePdefs.value;
                                                             makeSteps.value;
                                                             };

updatePdefs = { pdefs.do{| pdef | pdef.clear};
                                   pdefs = List[];
                                   presets[currentPreset].keys.do{| nombre |
                                        pdefMaker.value(nombre -> presets[currentPreset]
[nombre]);
                                   };
};

updateFileUsed = { | item |
                              selFile = item;
                              pathFile = "/Users/TAVO/Google Drive/Curso Intensivo SC/Proyecto
Final/Presets User/"++selFile++".rtf";
                              presets=Object.readArchive(pathFile);
                              arrayPresets = presets.order;
                              arrayPresets = arrayPresets.takeThese({ arg item; item.value ==
\Data; });
                              arrayPresets.do({ arg item, ind;
                                    if(item==\default,
                         {defPres = ind;},
                                                             {})
                                                              });
                              menPres.items_(arrayPresets.collect(
                                                        {|items| items}))
                                                        .value_(defPres);
                              updatePresets.value(\default);
};

// INSTRUMENT
instrument ={
var  promptSynth, sample;
promptSynth = 4000;
guiInstBeat=Window("Instrument", Rect(600,00,400,400)).front;
guiInstBeat.view.decorator = FlowLayout(guiInstBeat.view.bounds);
Button(guiInstBeat,  190@80)
        .states_([
            ["Synthdef", Color.black, Color.blue(0.7,0.2)],
            ["Synthdef", Color.black, Color.blue(0.7,0.8)]
            ])
            .action_({ |val| if(val.value==1,
                 (Document.open("/Library/Application
Support/SuperCollider/Extensions/BEAT/BEAT.sc",promptSynth,0);
```

```
                );
                );
            });
Button(guiInstBeat,  190@80)
        .states_([
            ["Load File", Color.black, Color.blue(0.7,0.2)],
            ["Load File", Color.black, Color.blue(0.7,0.8)]
            ])
        .action_({ |val|
             bufferInstrumentMaker.value;

          });

                    };

bufferInstrumentMaker = {
                    Dialog.getPaths({ arg paths;
                    paths.do({ arg path;
                                            var nombre, buf;

                                            nombre =
path.basename.splitext[0].asSymbol.postln;
                                            buf = Buffer.read(Server, path);
                                            presets[currentPreset].add(
                                                 nombre ->(
                                                      amps: Array.fill(maxNumSteps,
{ 0 }),

                                                      durs: Array.fill(maxNumSteps,

{ 1 }),

                                                      gain: 1,
                                                      buffer: buf,
                                                      synthdef: if (buf.numChannels == 2)
                                                                      { \BEATPlayStereo }
{ \BEATPlayMono } ;


                                            ));
                                            instruments.add(nombre);
                                            pdefMaker.value(nombre -> presets
                                    [currentPreset][nombre]);
            });
            makeSteps.value;
            },
            {
             "cancelled".postln;
            });
instBeat = 1;
};


//MIXER

mixer = {
var  viewMix, gridMix, rango;
guiInstMix=Window("Mixer", Rect(200,00,400,400));
```

```
viewMix = View();
viewMix.layout = HLayout();
rango= ControlSpec(0.00, 1, \lin, 0.01,0.5);

instruments.do{|nombre|
            EZSlider(viewMix,
                                10@200,
                                nombre,
                                rango,
                                {|ind|

               presets[currentPreset][nombre][\gain]= ind.value;
                presets[currentPreset][nombre][\gain].dbamp;
                                            },
                                layout:\vert

            );

                    };

                    gridMix.add (viewMix);
guiInstMix.layout_(
gridMix= GridLayout.rows(
[viewMix, rows:1],


 );
 ).front;
instMix = 1;
};


CmdPeriod.doOnce({guiMain.close;
"Chau ||BEAT https://github.com/vjtavo , contact: vjtavo@outlook.com (UNAM, CMMAS) MEXICO
2014".postln;
                    if(instMix==1,{guiInstMix.close},{});
                    if(instBeat==1,{guiInstBeat.close},{});
                                });

guiMain.onClose_({
                    var winInst, winMix;
                     pdefs.do{| pdef | pdef.clear };
"Chau ||BEAT https://github.com/vjtavo , contact: vjtavo@outlook.com (UNAM, CMMAS) MEXICO
2014".postln;

                    if(instMix==1,{guiInstMix.close},{});
                    if(instBeat==1,{guiInstBeat.close},{});
                    }
);

reset= {   guiMain.close;
             guiMain.onClose_({
                    var winInst, winMix;
                     pdefs.do{| pdef | pdef.clear };
```

```
                    beat.value;
                    });

                TempoClock.tempo = 80/60;
        };

};
beat.value;


}
}
```

Lista de Referencias

Adje, A. (2005). *Aerófonos mexicas de las ofrendas del recinto sagrado de Tenochtitlan. (*Tesis de doctorado). Universidad libre de Berlin. Berlin.

Alonso, R. (2009). Tu computadora es un campo de batalla. *Tensiones tecnológico-político-culturales en la era de las tic.* Córdoba: Ediciones del Centro Cultural España-Córdoba.

Amatriain, X. (2004). *An Object-Oriented Metamodel for Digital Signal Processing with a focus on Audio and Music.* Thesis. Universitat Pompeu Fabra. Barcelona.

Asuaga, C.(Coord.)(2011). *La cultura en Uruguay:*
una mirada desde las ciencias económicas. Fundación de Cultura Universitaria (FCU). Montevideo.

Bages, J. (2011, febrero). “Sobre las nuevas tecnologías musicales y los sistemas musicales interactivos”.[versión electrónica] *Espacio Sonoro,*No. 23,1-25.

Benjamin, W. (1973).La Obra de Arte en la Época de su Reproductibilidad Técnica Traducción de Jesús Aguirre. Ed. Taurus, Madrid.

Berenguer, X. (2002). “*Arte y tecnología: una frontera que se desmorona”.* Recuperado el 21 de febrero de 2013, http://anagangulo.es/index.php/tesis/41-textos/74-arte-y-tecnologia-una-frontera-que-se-desmorona.html

Berrón, V. (2007). Augusta Ada King, Condesa de Lovelace en *Revista Digital Matemáticas, Educación e Internet* Volumen 8, número 2 recuperado el 6 de mayo de 2012, http://www.cidse.itcr.ac.cr/revistamate/paginasgenerales/indexv8n22007.htm

Camacho, G. (2001). “Hacia una traducción de las culturas musicales” en *Traduic*, no. 16, año 9, Universidad Intercontinental, 4-7.

Cage, J. (1958). *El futuro de la música: Credo.* Recuperado el 13 de enero de 2012, http://www.ccapitalia.net/reso/articulos/johncage/futuromusica.htm

Cascone, K. (2000). The Aesthetics of Failure: 'Post-Digital' Tendencies in Contemporary Computer Music, *Computer Music Journal* 24, p. 12. Recuperado el 30 de marzo de 2012, http://www.leonardo.info/lmj/sherburnelmj13cdintro.html

Chowning, J. (1973). The Synthesis of Complex Audio Spectra by Means of Frequency Modulation.En *Journal of the Audio Engineering Society* 21, no. 7, 526-534.

Collins, N., McLean, A., Rohrhuber, J. & Ward, A. (2003) *Live coding in laptop performance* Cambridge University Press. Recuperado el 27 de agosto de 2010.

Doornbusch, P. (2012). *A Chronology / History of Electronic and Computer Music and Related Events 1906 – 2012.* En Roger T. Dean (Ed.). The Oxford Handbook of Computer Music. Recuperado el 27 de febrero de 2013 de http://www.doornbusch.net/chronology/#USAGE.

Galard, J. (2000). La computadora en la música. [versión electrónica]. *Revista Anales del Instituto de Investigaciones Estéticas.* Volúmen XXII, número 77, 263-267.

García de León, A. (2002) *El mar de los deseos. El Caribe hispano musical, historia y contrapunto*. México, Siglo XXI.

Gómez, D. (2006). *Programando con Small Talk*. Lin Editorial S.L.

Haro, J.(2006). La escucha expandida. *Cuadernos del Centro de Estudios en Diseño y Comunicación* [Ensayos] No 20. (pp. 41-48). Buenos Aires, Universidad de Palermo.

Horkheimer, M. y Adorno, T. (1988) *Dialéctica del iluminismo.* Buenos Aires, Sudamericana.

Hutchins, C. (2005). *SuperCollider Tutorial Chapter 4.* Creative Commons License: Attribution Only. Recuperado el 17 de febrero de 2011, de http://www.berkeleynoise.com/celesteh/code/tutorials/

Jáuregui, J. (1987). *El mariachi como elemento de un sistema folklórico*, en Palabras Devueltas, homenaje a Claude Levi-Strauss(pp. 93-126), México, INAH/CEMCA/IFAL.

Jórda, S.(2008). *Interactivity and live computer music.*En Nick Collins y Julio d'Escrivan. (Eds.). The Cambridge Companion to Electronic Music (pp. 89-106). Cambridge: Cambridge University Press.

Kelty, C. (2008). *Two bits The Cultural Significance of free Software.* USA: Duke University Press.

Lessig, L. (2005). *Cultura libre. Cómo los grandes medios usan la tecnología y las leyes para encerrar la cultura y controlar la creatividad*; Santiago de Chile, LOM Ediciones.

Machado, A. (2006). Arte y medios: Aproximaciones y diferencias. *Revista KEPES*, año 3 No. 2, 145-163.

Magnusson, T. (2002) *Processor Art.* Cand. Mag Thesis. University of Copenhagen.

Magnusson, T. (2011) Algorithms as Scores: Coding Live Music. *LEONARDO MUSIC JOURNAL*, Vol. 21, 19–23.

Manning, P. (2004). *Electronic and Computer Music.* Oxford: Oxford University Press.

Manovich, L. (2008). *Software Takes Command.*CC license.

Marchiaro, P. (2009). *¿Desea guardar los cambios?.* Córdoba: Ediciones del Centro Cultural

España-Córdoba.

Massarini, A. (2007). “Democratizar el conocimiento científico: criterios y estrategias para un cambio en la enseñanza de las ciencias” e*n IV Congreso Comunicación Social de la Ciencia, Madrid.* Recuperado el 13 de diciembre de 2012, http://www.csciencia2007.csic.es/actas/co_a3_01.pdf

Matthew Duignan, James Noble, Robert Biddle (2005) A TAXONOMY OF SEQUENCER USER-INTERFACES Ann Arbor, MI: MPublishing, University of Michigan Library.

McCartney, J. (1996). *SuperCollider: a new real time synthesis language.* Proceedings of the International Computer Music Conference 1996. Recuperado el 16 de enero de 2011, de http://quod.lib.umich.edu/cgi/p/pod/dod-idx/supercollider-a-new-real-time-synthesis-language.pdf

McCartney, J. (2002). *Rethinking the computer music language: SuperCollider.* Computer Music Journal, 26, 61–68.

McLean, A., Griffiths, D, Collins, N. y Wiggins, G. (2010) Visualisation of Live Code, *EVA London Conference Proceedings*, Londres recuperado el 10 de octubre de 2012, http://www.bcs.org/upload/pdf/ewic_ev10_s2paper1.pdf

Medina, L. (2010). *Software libre y arte: interacciones y plataformas.* Colombia. Recuperado el 12 de abril de 2011, www.bdigital.unal.edu.co/6007/1/luisfernandomedinacardona.pdf

Miyara, F. (1995). *La música por computadora.* Recuperado el 18 de enero de 2011, de la base de datos de la Subsecretaria de Cultura de la Provincia de Santa Fe. www.fceia.unr.edu.ar/acustica/biblio/musica-pc.pdf

Organización de las Naciones Unidas. (2008). Declaración Universal de los Derechos Humanos, United Nations. Recuperado el 10 de septiembre de 2012,

http://unesdoc.unesco.org/images/0017/001790/179018m.pdf

Orozco, J. (2007). *Etimologías griegas*. México: Pearson Educación.

Pajares, R. (2011). *Historia de la música en 6 bloques. Bloque 4*. Madrid: Editorial Visión Libros.

Piscitelli, A. (2005). *Internet: La imprenta del siglo XXI.* Barcelona: Editorial Gedisa.

Ruesga, J. (2005). Intersecciones híbridos y derivados. En Ruesga, J. (Ed.). *Intersecciones la música en la cultura electro-digital* (pp. 5-16). Sevilla: arte/facto, Colectivo Cultura Contemporánea. Area de Cultura del Ayuntamiento de Sevilla.

Small, C. (1989). *Música, sociedad, educación.* Alianza Editorial.

Sorensen, A. (2005). Impromptu: An interactive programming environment for composition and performance. En *Proceedings of the Australasian Computer Music Conference 2005*, pages 149–153. Recuperado el 27 de agosto de 2011 de http://eprints.qut.edu.au/31056/1/c31056.pdf

Soler, P. (2008). Artists and Free Software – an Introduction. *En FLOSS + Art .* Version 0.1. Recuperado el 10 de mayo de 2013, http://pcm.peabody.jhu.edu/~gdw/stdmp/docs/FLOSS+Art.pdf

Stallman, R. (2004). *Software libre para una sociedad libre.* Madrid: Traficantes de Sueños.

Valle A. (2011). *Audio physical computing.* CIRMA - Universita` di Torino. Recuperado el 11 de febrero de 2012 de http://academia.edu/1991426/Audio_Physical_Computing

Varela, D. (2005). “Sobre la música experimental en los inicios del nuevo Siglo” en Ruesga, J. (Ed.). *Intersecciones la música en la cultura electro-digital* (pp. 109-118). Sevilla: arte/facto,

Colectivo Cultura Contemporánea. Area de Cultura del Ayuntamiento de Sevilla.

Vercelli, A. (2004). *La conquista silenciosa del ciberespacio.* Argentina: Autoedición.

Vestfrid, P. y Echeverría, M. (2010). *Tridecaedro: jóvenes investigadores en Ciencias Sociales de la UNLP* Argentina: Universidad Nacional de la Plata.

Wang, G. (2008). A History of Programming and Music. En Nick Collins y Julio d'Escrivan. (Eds.),*The Cambridge Companion to Electronic Music* (pp. 55-71). Cambridge: Cambridge University Press.

Yúdice, G. (2009). *La web 2.0 y el pacto social.* Córdoba: Ediciones del Centro Cultural España-Córdoba.

Zadel M. y Scavone G. (2006). *Laptop Performance: Techniques, Tools, and a New Interface Design.* Recuperado el 11 de febrero de 2012 de http://www.music.mcgill.ca/~gary/papers/zadel_scavone_icmc2006.pdf