UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Programa de Maestría y Doctorado en Música

Facultad de Música
Centro de Ciencias Aplicadas y Desarrollo Tecnológico
Instituto de Investigaciones Antropológicas

Live coding en red
Una práctica de networking en el caso de LiveCodeNet Ensamble

TESIS
QUE PARA OPTAR POR EL GRADO DE
MAESTRO EN MÚSICA (TECNOLOGÍA MUSICAL)

PRESENTA
Hernani Villaseñor Ramírez

TUTOR PRINCIPAL
Jesús Fernando Monreal Ramírez
(Escuela Nacional de Conservación, Restauración y Museografía, INAH)

CIUDAD DE MÉXICO. NOVIEMBRE 2017

Declaro conocer el Código de Ética de la Universidad Nacional Autónoma de México, plasmado en la Legislación Universitaria. Con base en las definiciones de integridad y honestidad ahí especificadas, aseguro mediante mi firma al calce que el presente trabajo es original y enteramente de mi autoría. Todas la citas de, o referencias a, la obra de otros autores aparecen debida y adecuadamente señaladas, así como acreditadas mediante los recursos editoriales convencionales.

Agradecimientos

A LiveCodeNet Ensamble: Emilio, Libertad, José Carlos, Eduardo y Katya quienes han compartido en diferentes momentos esta forma de hacer música, y quienes durante la investigación estuvieron disponibles para contestar mis preguntas.

A mi compañera Sabine, por todo el apoyo recibido durante la realización de este trabajo. A mis padres Josefina y Heriberto por su apoyo incondicional. A mis hermanas Aída, Gilda, Isolda y Luisa. A mis sobrinas Frida y Elisa y a su papá Adolfo.

A Fernando Monreal por aceptar el reto de dirigir esta tesis, compartirme libros y cuestionarme de manera crítica para provocar ideas durante el proceso del trabajo.

A mis compañeros y compañeras de maestría, en especial a quienes conformaron el Seminario de Investigación: Jorge David, Mercedes, Edwin y Gerardo que con sus lecturas ayudaron a darle forma a las primeras versiones. A Fabián por las lecturas de este trabajo y el texto del coloquio.

Al Centro Multimedia mi anterior lugar de trabajo donde aprendí a hacer live coding y donde se gestaron los primeros planteamientos de la tesis.

A los lectores y lectoras: Dr. Rodrigo Sigal Sefchovich, Dr. Jorge David García Castilla, Mtra. Cinthya García Leyva, Dra. Rossana Lara Velázquez y Dr. Hugo Solís Gracía por su tiempo y lectura dedicada que potenciaron las ideas del texto desde diferentes perspectivas.

A mis profesores y profesoras de la maestría quienes propiciaron la construcción de nuevos conocimientos en torno a la tecnología musical y el sonido.

Al Programa de Maestría y Doctorado en Música de la UNAM encargado de coordinar el proceso que conlleva un proyecto de esta naturaleza. Al Programa de Becas para Estudios de Posgrado en la UNAM, por el apoyo recibido durante tres semestres y al Programa de Apoyo a los Estudios de Posgrado de la UNAM, por el préstamo de equipo.

Índice

Introducción

Este trabajo responde a la inquietud por entender cómo se configura la práctica musical realizada por un grupo de músicos en una *red de computadoras*, mediante la escritura de *código fuente* para generar sonido, durante un concierto o ensayo en el contexto del *live coding*. Entiendo por *live coding* la práctica de escribir código fuente con la intención de crear música en el momento; y por *red* la relación que hay entre la infraestructura tecnológica y la *red de relaciones* creada por los integrantes del grupo en el momento de realizar dicha práctica. LiveCodeNet Ensamble[1] (LCNE) es un grupo mexicano que crea música con estas características del cual soy integrante desde su formación en octubre de 2013. He tomado su caso para ilustrar el objeto de estudio definido por la práctica musical con computadoras antes mencionada a la cual me refiero en este trabajo como *live coding en red*.

El objeto de estudio involucra aspectos técnicos y sociales que se relacionan en el momento de poner en marcha esta práctica. Los aspectos técnicos que analizo en el presente trabajo están acotados a dos tecnologías para realizar live coding en red: el lenguaje de programación y la red de computadoras. En el primer caso, un lenguaje de programación que permite escribir y modificar código fuente en el momento de un concierto o ensayo; en el segundo caso, una red de computadoras con un diseño local, topología de estrella, par a par y de difusión que permite a un grupo comunicarse, intercambiar, compartir y sincronizarse en un mismo lugar[2].

Lo anterior puede ser enmarcado bajo el concepto de *networking* de Tatiana Bazzichelli (2008; 2013), el cual delinea las prácticas de networking sugeridas por la autora

1 Para una referencia amplia de LiveCodeNet Ensamble véase <https://livecodenetensamble.wordpress.com/>

2 Este trabajo se enfoca en el estudio de una práctica de live coding en red realizada con la presencia de los integrantes en el mismo lugar contrario a una que se realiza con una distribución de los participantes a la distancia. El término local proviene de la configuración de la red de computadoras y hace referencia a esta condición.

como aquellas en las que se construye una red de relaciones en el momento de realizar una práctica de creación colectiva dentro una infraestructura de red —por ejemplo, el correo postal o una red de computadoras— a través de la que se comparten e intercambian conocimientos, ideas y experiencias; y en la que se crea un mapa de conexiones en progreso. Encuentro conveniente utilizar este concepto debido a su perspectiva de análisis sobre las prácticas colectivas que estudia desde un contexto tecnológico y social.

A partir de lo antes descrito, los objetivos de la tesis son: analizar y describir la práctica musical de live coding en red atendiendo los aspectos técnicos de la red de computadoras y el lenguaje de programación; y los aspectos sociales descritos en el concepto de networking. Definir el vínculo entre live coding en red y las prácticas de networking. Analizar y describir la práctica de LiveCodeNet Ensamble bajo los aspectos antes mencionados.

Las preguntas que guían la investigación son ¿de qué manera se configura la práctica de live coding cuando es realizada en grupo con el uso de una red de computadoras?, ¿cómo se vincula el live coding en red a las prácticas de networking?, ¿qué aspectos técnicos y sociales están involucrados y cómo se relacionan? y ¿de qué manera se configuran en el caso de LCNE? Para abordar las preguntas parto de la siguiente hipótesis: si consideramos al live coding en red una práctica de improvisación colectiva que se realiza a través de una infraestructura tecnológica, como la red de computadoras, es posible considerarlo una práctica de networking pues al improvisar se crea una red de relaciones en el momento de construir el sonido grupal.

Con la finalidad de contestar las preguntas de investigación y desarrollar los objetivos planteados, el trabajo establece una serie de conexiones que relacionan al live coding en red con la noción de networking a través de diferentes ideas y conceptos que trazan una línea entre las prácticas de *live coding*, *improvisación libre* y *network music; y* los movimientos *hacker*, *open source* y *software libre.* En esta serie de asociaciones los términos *compartir* e *intercambiar* establecen un puente entre la red de computadoras y la red de relaciones sociales. Así mismo, el trabajo consiste en el análisis de la práctica

musical realizada por el grupo de live coding LiveCodeNet Ensamble y en el estudio de la red de computadoras que utiliza. El análisis de la práctica del ensamble se llevó a cabo durante el periodo comprendido por la maestría a partir de entrevistas con sus integrantes, grabaciones de audio y video de conciertos y ensayos, así como mi participación en el ensamble[3].

La tesis consta de tres capítulos y está dividida de la siguiente manera: en el primer capítulo planteo un panorama del live coding a partir de varios autores (McLean 2011; Rohrhuber y De Campo, 2009; Ward et al. 2004) enmarcándolo como práctica de creación musical ligada a la improvisación con base en diferentes discursos de la improvisación libre (Bailey, 2010; Berenguer, 2007; Matthews, 2012; Moraes Costa, 2015). Para entender lo anterior, expongo el concepto de código fuente y cómo es posible generar sonido en el momento de un concierto o ensayo a partir de éste. Además, discuto la figura del practicante de live coding o live coder (Di Próspero, 2015; McLean, 2011) en el contexto de la música y la improvisación.

El segundo capítulo trata acerca de la red de computadoras (Barceló et al. 2004; Tanenbaum y Wetherall, 2012) y su uso en la música a partir de la práctica de *network music* (Brown y Bischoff, 2002; Rohrhuber, 2007). En este capítulo expongo también el concepto de *networking* (Tatiana Bazzichelli, 2008; 2013) y bajo este marco discuto la práctica de live coding en red a partir de la serie de asociaciones mencionadas.

El tercer capítulo se centra en el análisis y descripción de la práctica de live coding en red realizada por LiveCodeNet Ensamble, así como en el análisis y descripción de la red

3 El periodo de la maestría en el que realicé la investigación comprende agosto 2014 a mayo 2016. En ese lapso LCNE realizó siete conciertos y una residencia virtual; de esas actividades analicé cinco conciertos y sus ensayos comprendidos entre mayo 2014 y enero 2016 con base en grabaciones de audio y/o video <https://livecodenetensamble.wordpress.com/fechas/>. También, dentro del periodo mencionado realicé dos entrevistas con cada integrante del ensamble; algunas se llevaron a cabo después de los ensayos, otras se realizaron fuera de éstos o por correo electrónico. En todos los casos realicé entrevistas semiestructuradas, partiendo de las mismas preguntas para todos los entrevistados. Cabe mencionar que, como integrante del ensamble, participé en todos los conciertos.

y el software que utilizan. En este capítulo establezco un diálogo entre los procesos que realiza el ensamble, los conceptos definidos en los capítulos anteriores y la serie de entrevistas realizadas con sus integrantes. El capítulo también contiene una discusión acerca de la sincronía y la apertura del código fuente en el momento que el ensamble realiza live coding.

Por último, es pertinente aclarar algunos criterios que he seguido en la redacción del texto. En cuanto a las cursivas, este trabajo reconoce como anglicismos los términos *live coding*, *live coder*, *networking, network music* y *open source*, sin embargo, no siempre aparecen en cursivas ya que sigo la convención propuesta por Chiantore, Domínguez y Martínez (2016) de escribir así las palabras extranjeras que forman parte de un vocabulario común en la tradición musical estudiada. En lo subsecuente mantengo el criterio propuesto por los mismos autores de utilizar cursivas para enfatizar términos relevantes, palabras extranjeras —a excepción de las antes mencionadas— y títulos de obras.

Las citas que provienen de otros idiomas las he traducido al español. El criterio que sigo es colocar las citas traducidas en el cuerpo del texto en tanto que las citas en su idioma original están situadas como nota a pie de página. Las citas que son escritas de manera directa a pie de página también están traducidas, en este caso el texto en idioma original lo coloco al lado entre corchetes. Los ejemplos de código computacional están escritos con el programa SuperCollider y utilizo una fuente distinta a la del cuerpo del documento; en ocasiones escribo palabras con esta fuente en el texto principal para indicar que pertenecen al lenguaje de programación, por ejemplo, `doneAction`.

1. Live coding

> Code does not always nor automatically do what its says, […]
> —Wendy Chun, *On "Sourcery," or Code as Fetish*

Este capítulo busca comprender qué es el live coding, dentro del campo de conocimiento de la música por computadora, definiéndolo a partir del código fuente, el practicante de live coding o live coder y la improvisación libre. Para entender esta relación, el capítulo comienza planteando un panorama de la práctica de live coding, a continuación discute qué es el código fuente y cómo se produce sonido a partir de éste, después la figura del live coder y por último la relación con la improvisación. En esta sección del trabajo se plantea la práctica del live coding desde la individualidad y se establece desde un base teórica que sirve para argumentar, en los siguientes capítulos, cómo ocurre de manera grupal y en una red.

Dentro del campo de la música por computadora el *live coding* es una práctica de creación musical relacionada con la *improvisación*. La forma en que se realiza es escribiendo *código fuente* en el momento para crear música durante un concierto o ensayo. El código fuente es un texto escrito en un lenguaje de programación[4] que contiene una serie de instrucciones dirigidas a una computadora para indicarle qué tareas o acciones realizar, por ejemplo, generar sonido. En el live coding, este texto se escribe sobre la marcha para actualizar de manera continua las instrucciones que la computadora va convirtiendo en sonido que cambia conforme el código modificado es ejecutado.

Según Alex McLean (2011), el término *live coding* aparece alrededor del año 2003

4 Según Norton (2014), los lenguajes de programación son una variante básica del idioma inglés que "le permiten al programador describir un programa" (p. 502). Por otro lado, el autor dice que el contenido del código son "declaraciones escritas" (p. 501) o instrucciones para que la computadora realice tareas.

para definir “la actividad de un grupo de practicantes e investigadores que comenzaron a desarrollar nuevos enfoques para hacer música con computadora y video animación”[5] (p. 129). Julian Rohrhuber y Alberto de Campo (en Collins et al., 2011) se remontan a los años sesenta para situar un antecedente en prácticas experimentales de programación de las ciencias de la computación y las matemáticas experimentales que eventualmente darían lugar al live coding.

Los nuevos enfoques en la creación musical a los que se refiere McLean, en la cita del párrafo anterior, se desarrollan a partir de la escritura de código fuente y su modificación en vivo para obtener sonido. En este sentido, Collins, McLean, Rohrhuber y Ward (2003) mencionan que los practicantes de live coding parten del reto de “codificar música al vuelo”[6] (p. 321), de escribir programas mientras tocan con ellos, lo cual conlleva riesgo e improvisación. Al respecto, Rohrhuber y De Campo (en Collins et al., 2011) mencionan que para el live coding “el programa es una reflexión del pensamiento y una parte integral del proceso de razonamiento”[7] (p. 207) lo que hace de esta manera de escribir código un paradigma “relevante para la música improvisada”[8] (ibíd.).

El presente trabajo enmarca al live coding dentro de un contexto sonoro, sin embargo, cabe mencionar que esta práctica se realiza en diferentes disciplinas. Al respecto, Thor Magnusson (2013) dice que “con base en el performance musical, el live coding ahora se ha hecho común en artes visuales, sistemas de luces, robótica, danza, poesía y en otras formas de arte que pueden operar con instrucciones algorítmicas”[9] (párr. 1). A su vez, el live coding puede tener distintos objetivos como la didáctica en la enseñanza de programación (Rohrhuber y De Campo, 2009; Bell, 2014) o de música con código. Por otra

5 “[…] the activity of group of practitioners and researchers who had begun developing new approaches to making computer music and video animation”.

6 “coding music on the fly”.

7 “the program is a reflection of thinking and an integral part of the reasoning process”.

8 “This paradigm is relevant for improvised music”.

9 “With its foundation in musical performance, live coding has now become common in visual arts, light systems, robotics, dance, poetry, and other art forms that can operate with algorithmic instructions”.

parte, Blackwell, McLean, Noble y Rohrhuber (2013) mencionan que el live coding se estudia desde diferentes campos de conocimiento como las humanidades, la educación en computación y la ingeniería de software.

Diferentes definiciones del live coding involucran improvisación, creación musical, programación, escritura de código en el momento y mostrar el proceso de escritura a la audiencia. Una de las primeras definiciones, y quizá la más referida, es la propuesta por Ward et al. (2004), quienes definen esta práctica de la siguiente manera: “live coding es la actividad de escribir (partes de) un programa mientras está corriendo”[10] (p. 243). Los autores enuncian esta definición desde de un contexto artístico y expresan que un programa informático —que genera sonido— puede crearse mientras está en funcionamiento; agregan que esto difumina la división entre herramienta y lo que ésta produce. No obstante, la anterior definición no aporta suficiente información para entender qué es live coding, sobre todo si se piensa desde la creación musical. Quizá la siguiente respuesta, obtenida por McLean (2011) en una encuesta realizada a varios practicantes de live coding[11], puede complementar la definición para entenderla en un contexto musical: “construir tu instrumento mientras lo ejecutas”[12] (p. 167).

La anterior respuesta es una metáfora de cómo un *live coder*[13] genera sonido mediante la escritura y modificación de código fuente que va conformando un programa al mismo tiempo que la computadora lo ejecuta para producir música y sonido. La analogía entre código e instrumento es recurrente en la teoría del live codig, por ejemplo, Zmölnig y Eckel (2007) dicen que el live coding es una técnica de programación con la que se escribe un instrumento en forma de software, mientras que Wang, Mirsa, Davidson y Cook (2005)

10 “Live coding is the activity of writing (parts of) a program while it runs”.

11 La lista de definiciones puede consultarse en el Apéndice A de la Tesis de doctorado *Artist-Programmers and Programming Languages for the Arts* escrita por Alex McLean.

12 “building your instrument while playing it”.

13 Por live coder me refiero al practicante de live coding. Para una discusión acerca del término véase la sección 1.1

consideran al código como un "instrumento musical expresivo"[14] (párr. 3) utilizado en el acto de programar música en vivo. Este instrumento se puede pensar como código que contiene la descripción de una fuente sonora y las instrucciones de cómo accionarlo para ejecutarlo. Este concepto se ve reflejado en el desarrollo de algunos programas diseñados para realizar live coding como Sonic Pi que se define como un sintetizador de live coding musical[15].

Por otro lado, Di Próspero (2015) dice que "el live coding […] es una expresión artística en la cual los live coders generan música algorítmica en vivo con una laptop o computadora portátil, escribiendo líneas de código con diferentes lenguajes de programación" (p. 46). Mientras que Blackwell et al. (2013) lo definen como "programación interactiva improvisada, básicamente para crear música electrónica y otros medios digitales, hecha en vivo con una audiencia"[16] (p. 130).

Además de la palabra live coding, otros términos son utilizados para describir la escritura de código en el momento para generar sonido. Ge Wang y Perry R. Cook (2004) proponen el término *on-the-fly programming* que definen como una forma de programar en la que se "aumenta y modifica un programa mientras está corriendo"[17] (párr. 1). Como es posible observar, la definición *on-the-fly programming* de Wang y Cook es parecida a la de *live coding* propuesta por Ward et al. (2004): ambas se refieren a una práctica de programación en la que el programa opera mientras es escrito. Esta idea es objetada por Rohrhuber y De Campo (en Collins et al. 2011) quienes afirman que "es imposible escribir un programa mientras corre"[18] (p. 207), en cambio, plantean que es posible organizar el código de un programa de tal forma que ciertas partes se intercambian de forma dinámica y pueden reescribirse mientras que todo el proceso continua, lo que permite pensar el código

14 "expressive musical instrument".

15 El sitio web de Sonic Pi presenta el siguiente eslogan: "The Live Coding Music Synth for Everyone".

16 "[…] improvised interactive programming, typically to create electronic music and other digital media, done live with an audience".

17 "augments and modifies the program while it is running".

18 "It is impossible to write a program while it runs".

fuente como la principal interfaz. A esta forma de escribir un programa la llaman *Just-in-Time programming*. Además de este término, los autores, utilizan *programación conversacional* o *interactiva* para referirse al live coding; la idea proviene, según Rohrhuber y De Campo, de que los live coders utilizan el código como un medio conversacional en el momento que escriben composiciones algorítmicas durante un concierto.

El live coding involucra la escritura de instrucciones en forma de código fuente para que una computadora realice acciones; en el caso de este trabajo, acciones que generan sonido. Una escritura que se realiza en el momento con base en decisiones estéticas. Los siguientes apartados desarrollan la relación que hay entre la escritura y modificación de código fuente, las acciones que realiza una computadora y el sonido que resulta de dichas acciones. Además, se discute el rol del practicante de live coding, el término código fuente y el vínculo que hay entre el live coding y la improvisación.

1.1 Live coder: practicante de live coding

Para referirme al practicante de live coding, en el presente trabajo, utilizo la palabra *live coder*, término empleado en la teoría de esta práctica por diferentes autores. Por ejemplo, Alex McLean (2011) dice que los live coders son "músicos con computadora quienes escriben música en el dominio digital bajo tiempos estrictamente reducidos"[19] (p. 63). McLean se refiere a individuos que crean música mediante la escritura de código fuente en un tiempo determinado por la duración de su participación en un concierto. Es interesante observar que McLean se refiere al live coder como músico y no como programador; el autor le atribuye una decisión artística más que una decisión técnica en el momento que escribe código.

19 "computer musicians who notate music in the digital domain under tight time constraints".

Por su parte, la antropóloga social Carolina Di Próspero (2014) define al live coder en cuanto músico y programador de la siguiente manera:

> Ambos roles tienen características particulares, por ejemplo, no son músicos electrónicos como los que estamos más acostumbrados a ver y oír, tampoco son simples programadores de software, hay una diferencia de intención que tiene que ver con explorar más que ejecutar, crear más que reproducir, experimentar el proceso más que alcanzar el producto terminado. (p. 47)

Di Próspero se refiere a músicos que escriben código con la intención de improvisar[20]. En este sentido, se puede definir al live coder como improvisador o improvisadora que crea música a través de la escritura y modificación de código fuente con el que interviene un proceso musical y computacional de forma continua para generar cambios en el sonido que produce una computadora.

En lo antes expuesto, concuerdo con McLean y Di Próspero quienes definen al live coder como músico e improvisador bajo las condiciones de la creación musical con computadora a través de la escritura de código fuente. Al respecto, hago la aclaración que el término músico lo utilizo en este trabajo de manera amplia y no acotado a un individuo con formación especializada.

Sin embargo, en lo antes mencionado hay una contradicción pues aunque considero que, por un lado, el live coder no necesariamente tiene una formación musical, aunque por el otro, requiere conocimientos de programación para llevar a cabo esta práctica. En este sentido, la figura de live coder presenta la dicotomía de músico-programador. Al respecto, Sarah Bell (s.f) dice que “los live coders vienen de dos direcciones —son artistas que quieren aprender a programar, y programadores que quieren

20 Para una discusión acerca de la relación entre improvisación y live coding véase la sección 1.4.

expresarse”[21] (párr. 3). Yo agregaría que hay un interés por ambas cosas y que no necesariamente son artistas o programadores. En este sentido, el live coder se define en la combinación de la programación y la música, como individuo que toma decisiones estéticas en el momento que escribe código con la intención de crear música.

Respecto al término músico, Wade Matthews (2012) apunta que es impreciso pues describe tanto al creador de música como al que se dedica a reproducirla. A partir de esta acotación, Matthews plantea que los términos compositor e improvisador definen al creador de música aunque se distinguen por las técnicas que utiliza cada uno, el primero crea música en “tiempo diferido” (p. 19), el segundo lo hace en el momento. Por otro lado, Caleb Kelly (2009) argumenta que la música occidental se volvió más ruidosa durante el siglo XX, y que la distinción entre sonido musical —producido por instrumentos tradicionales tocados por músicos entrenados— y no musical se hizo menos marcada. A partir de esto, el autor dice que “en este momento de la historia de la música siento que es hora que el término “música” sea retomado por practicantes contemporáneos y usado sin preocuparse por su reciente pasado cuestionable en manos de la academia”[22] (p. 17). También comenta que se siente cómodo al utilizar la palabra “música” en prácticas que atraviesan la música, el sonido y el ruido. Yo incluiría al live coding como práctica de creación musical contemporánea que absorbe estos referentes.

1.2 Código fuente: lo que se escribe en el live coding

En las secciones anteriores se menciona que el live coding se realiza a partir de la escritura

21 “Live coders come from two directions — artists who want to learn to code, and coders who want to express themselves”.

22 “At this point in the history of music I feel it is high time that the term “music” be taken back by contemporary practitioners and used without concern for its recent shaky past in the hands of the academy”.

de código fuente. Pero ¿qué significa código dentro del live coding? La palabra código se usa en distintos contextos y puede referirse a la encriptación para obtener una escritura secreta, a leyes, a métodos de criptografía o a la escritura de instrucciones computacionales (Cramer, 2005; Kittler, en Fuller, 2008); es el último uso el que tiene interés para el presente trabajo.

Según Florian Cramer (2005), el código computacional es la transformación de símbolos en acciones a partir de reglas e instrucciones escritas en forma de algoritmos[23] y puede definirse como una "secuencia de instrucciones ejecutables"[24] (p. 9). La transformación de símbolos en acciones planteada por Cramer (2005) pasa de un tipo de código a otro a partir de una traducción que convierte el código fuente en código máquina[25]. El primero lo escribe, lee y entiende un humano; el segundo lo entiende la computadora como resultado de una traducción que realiza[26].

El código fuente está relacionado con los conceptos de programa y lenguaje de

23 Según Andrew Goffey (en Fuller, 2008) un algoritmo es una entidad abstracta, la cual, al hablar de computación, puede decirse que es "un proceso efectivo para resolver un problema" (p. 16) [an effective process for solving a problem], lo cual interpreta, en términos de Alan Turing, como una serie de instrucciones que han sido introducidas a una computadora para resolver dicho problema. Por su parte, Glenn Brookshear (2012) define de manera informal el concepto de algoritmo como "un conjunto de pasos que define cómo llevar a cabo una tarea" (p. 224), posteriormente, el autor propone la definición formal como "Un algoritmo es un conjunto ordenado de pasos ejecutables y no ambiguos, que definen un proceso finito con un fin determinado" (p. 223). Para un desarrollo más amplio del concepto de algoritmo véase Algoritmos en *Introducción a la computación*, Brookshear, G. (2014).

24 "sequence of executable instructions".

25 Norton (2014) dice que el código máquina utiliza el sistema binario de unos y ceros, lenguaje que entienden las computadoras. "Debido a que estos unos y ceros forman el lenguaje del hardware de computadoras, este código se conoce normalmente como código máquina o lenguaje de máquina" (p. 502).

26 Según Brookshear (2012), *traducción* en computación se refiere al "proceso de convertir un programa escrito en un lenguaje de alto nivel a un formato ejecutable para la máquina" (p. 315). El autor dice que el proceso de conversión pasa de un "programa fuente" a un "programa objeto" en tres pasos: "análisis léxico", "análisis sintáctico" y "generación de código" (ibíd.).

programación; según Chun (2008), la palabra software se emplea en lugar de código mientras que los términos lenguaje, código y software se confunden. La autora dice que la palabra software hace borrosa la diferencia entre el código legible para el humano, la interpretación legible para la máquina y lo que ésta ejecuta. Por su parte, Ge Wang (2009) dice que "un *programa* es una secuencia de instrucciones para una computadora"[27] (p. 55) y que un *lenguaje de programación* consiste en un conjunto de reglas sintácticas y semánticas que definen dichas instrucciones; a su vez, menciona que el lenguaje de programación puede realizar un traducción de los programas que escribe un humano a instrucciones que ejecuta una computadora.

Esta traducción es necesaria ya que una computadora no trabaja de manera directa con el código fuente, necesita convertirlo a un tipo de código que le permita ejecutar las acciones inscritas en él. La traducción de código fuente se puede realizar por alguno de los siguientes métodos: compilación o interpretación. Ambos procesos son realizados, según Peter Norton (2014), por un programa dedicado llamado *compilador* o en su caso *intérprete*. Al respecto, Mauro Herrera (2013) hace una distinción entre *lenguajes compilados* y *lenguajes interpretados*.

Norton menciona que "el compilador convierte todo el código fuente en código máquina y crea un archivo ejecutable" (p. 503); según el autor, una vez compilado, el código se vuelve un archivo ejecutable independiente de su compilador. Así mismo, menciona que cada lenguaje de programación requiere de su propio compilador para traducir código. Esto es mencionado por Herrera cuando dice que los lenguajes compilados "toman todo el código de un programa, lo traducen a lenguaje de máquina y nos entregan un archivo ejecutable" (2013, p. 32).

En cuanto al intérprete Norton señala que, al igual que el compilador, traduce el código fuente en código máquina, solo que en lugar de "crear un archivo de código de objeto ejecutable, lo traduce y luego ejecuta cada línea del programa, una a la vez" (ibíd.). Herrera menciona que este tipo de lenguajes "permiten la interpretación inmediata de su

27 "A *program* is a sequence of instructions for a computer".

código. Cuando se escribe un comando este puede ser traducido a lenguaje de máquina y ejecutado sin la necesidad de conocer el resto del programa" (ibíd.).

Un compilador traduce todo el código de un programa, en tanto que el intérprete puede traducirlo por fragmentos, esto deriva en que un programa escrito en lenguaje interpretado puede continuar funcionando mientras una de sus partes es actualizada. En este sentido, Norton dice que "los intérpretes traducen el código sobre la marcha" (2014, p. 503). Esta característica hace que los lenguajes interpretados sean los más convenientes para realizar live coding. Si retomamos la definición de live coding propuesta por Ward et al. (2004) —escribir un programa mientras corre— se puede decir que los autores se refieren a un lenguaje interpretado; de hecho, en el mismo artículo, plantean que la exploración del código en el live coding resulta más natural "a través de un lenguaje de programación textual interpretado"[28] (p. 243). No obstante, algunos lenguajes compilados también son utilizados para realizar live coding y aprovechan las pequeñas interrupciones que se crea al compilar para incorporarlas al resultado sonoro[29].

Lenguajes compilados	Lenguajes interpretados
Construyen un archivo ejecutable	Interpretan código línea por línea
Traduce con un programa compilador	Traducen con un programa intérprete
Traducen todo el archivo de código	Traducen sobre la marcha

Tabla 1. Diferencias entre lenguaje compilado y lenguaje interpretado

Por su parte, Wendy Chun (2008) menciona que el código, al ser interpretado o compilado, es traducido en comandos legibles para la computadora que luego son

28 "Such algorithmic fine detail is most naturally explored through a textual interpreted programming language".

29 Por ejemplo, Julio Zaldívar ha utilizado el lenguaje C para hacer live coding sonoro desde un microcontrolador ATMEL. En el siguiente vinculo se puede apreciar un breve fragmento de video en el que Zaldívar escribe código fuente en un lenguaje compilado: <https://vimeo.com/55453484>

ejecutados. Sin embargo, la autora dice que el código no es la fuente sino hasta que se ejecuta y que la posibilidad de ejecutarse lo distingue como lenguaje. Para la autora el código fuente es la sustitución de una palabra por una acción; el código fuente es resultado de una acción que se convierte en lenguaje.

Para ilustrar el código fuente en su forma textual, en la figura 1 presento un fragmento de código escrito con el programa SuperCollider[30]. Este fragmento contiene una instrucción que al ser traducida y ejecutada por la computadora genera cuatro notas musicales de manera aleatoria, una después de la otra, con un intervalo de un segundo de separación entre cada una. En este caso, el código fuente, en forma de texto, está ahí fijo, estático, describe una acción potencial.

```
Tdef(\notas, {
  4.do{
  {Saw.ar(rrand(60, 72).midicps, Line.kr(0.5, 0, 0.5, doneAction:
2))}.play;
  1.wait};
}
).play
```

Figura 1. Código escrito en el lenguaje SuperCollider con la instrucción de tocar cuatro notas

Algo que puede observarse en el ejemplo es la legibilidad de algunas partes del código, pues aún sin conocer el lenguaje de programación en el que está escrito se leen las palabras *do*, *line*, *doneAction*, *play* o *wait*, de las cuales algunas definen una acción. Chun (2008) comenta que la legibilidad del código fuente se debe a que sus comandos están basados en el idioma inglés por lo que es posible leer y entender lo que habrá a la salida. Sin embargo, esta legibilidad es de manera parcial pues requiere entender, además del

30 Para una discusión acerca del uso de SuperCollider en este trabajo véase 3.2.3.

lenguaje de programación, la lengua inglesa. El código del ejemplo anterior se puede leer de la siguiente manera:

```
Realiza la siguiente tarea cuatro veces:
 1 Escoge un número entre 60 y 72
 2 Conviértelo a una nota musical de la escala cromática
 3 Haz que suene cada segundo con una duración de medio segundo
 4 Repite los pasos anteriores
```

Figura 2. Lectura del código anterior

Hay que aclarar que el sonido que genera la computadora, cuando compila o interpreta, crea la sensación que el código fuente actúa de manera directa. Sin embargo, no es así; como se menciona, la computadora primero debe traducirlo. Es decir, el código fuente no es sonido y tampoco lo produce, al menos no de manera directa, más bien describe una acción potencial, una instrucción que la computadora deberá traducir para llevarla a cabo.

Lo anterior puede contrastar con lo que plantea Inke Arns (2005), quien retoma la teoría de actos de habla[31] y, desde ésta, le atribuye al código una performatividad que le otorga "la habilidad de realizar y ejecutar" (p. 6). La autora cataloga el código de programa como un acto de habla inlocutivo en el que *lo dicho* coincide con *lo hecho*; bajo esta comparación, Arns dice que el código no es descripción o representación, más bien se convierte en acción.

En este sentido, el live coder decide qué acciones realizará la computadora mediante las instrucciones que escribe, sin embargo, la acción que ejecuta la computadora ocurre en un tiempo distinto al de su escritura. El código escrito en vivo y la acción

31 Según Arns (2005) esta teoría es expuesta por J.L. Austin en su texto *How to do things with words* (1955), en la que las expresiones lingüísticas se utilizan para realizar o ejecutar actos, es decir, el habla es acción.

ejecutada no suceden de manera simultánea, hay una falsa simultaneidad entre ambos procesos[32]. Es decir, el código que vemos aparecer en la pantalla cuando el live coder lo escribe no es el mismo que se está ejecutando, suceden en un orden de secuencia de escritura acción. La palabra *mientras*, en la definición de live coding, introduce el factor de tiempo que trascurre y separa ambas acciones: la de escribir y ejecutar.

Algo que se observa, a partir del planteamiento desarrollado en esta sección, es que la escritura de código tiene un papel relevante en la práctica de live coding por lo que se puede decir que la relación del live coder con las acciones que busca generar a través de su computadora es textual. En este aspecto, Alexander Galloway (2004) comenta lo siguiente: “considero que las computadoras son fundamentalmente un medio textual. La razón es obvia: las computadoras están basadas en un lenguaje tecnológico llamado código”[33] (pp. xxiii-xxiv). Galloway menciona que el código, como lenguaje, crea vínculos entre disciplinas a partir del texto. Es desde este sentido de textualidad que el código se aborda en el presente trabajo; un texto con el que el live coder expresa las ideas que van surgiendo en el momento para crear música y enmarca al código fuente como escritura humana que hace posible la manipulación de una computadora en vivo con fines artísticos.

Las siguientes secciones del capítulo describen cómo, a partir de la escritura y modificación de código fuente y su posterior ejecución, se puede generar e interactuar con el sonido que produce una computadora de manera continua. También plantea que la escritura de código fuente, en el live coding, posibilita la creación de música en el momento de un concierto o ensayo, y cómo se relaciona con la improvisación musical.

32 El término *falsa simultaneidad* lo tomo de los comentarios al trabajo realizados por Cinthya García Leyva.

33 “I consider computers to be fundamentally a textual medium. The reason is obvious: computers are based on a technological language called code”.

1.3 Escribir código para generar sonido

Desde el momento que es posible escribir código y obtener un resultado sonoro sin que el proceso computacional que lo genera sea interrumpido cuando el código es compilado, el flujo continuo de sonido puede manipularse en función de instrucciones que se escriben y modifican en el momento. Desde los años cincuenta, varios desarrollos tecnológicos han reunido las condiciones para llegar a este punto. Se puede mencionar el procesador de transistores, el compilador, los lenguajes interpretados y los lenguajes de alto nivel.

El procesador de transistores es una innovación que, según DiBona, Okcman y Stone (1999), sustituyó las tarjetas lógicas perforadas aumentando la capacidad de procesamiento de las computadoras y propiciando su baja de precio. En la música por computadora esto se ve reflejado en lo que comentan Rohrhuber y De Campo (2009), quienes dicen que en cierto momento fue posible correr programas complejos en computadoras económicas y que desde entonces se han podido considerar un instrumento de improvisación musical.

Desarrollo del procesador de transistores
Invención del compilador
Lenguajes de programación interpretados
Lenguajes de programación de alto nivel

Tabla 2. Desarrollos tecnológicos que permiten realizar live coding

Otros desarrollos que contribuyen son el compilador y los lenguajes de alto nivel[34].

34 Según Peter Norton (2014), un lenguaje de alto nivel tiene una sintaxis parecida a un idioma humano, las instrucciones escritas contienen palabras familiares para los programadores, quienes “pueden leer, escribir y entender programas de computadoras de manera mucho más sencilla” (p. 519). Aunque, el autor comenta que las instrucciones se traducen a lenguaje máquina para que una computadora las entienda y realice.

Según Gerard O'Regan (2013), el concepto de compilador fue ideado por Grace Hopper, pionera de la computación, a finales de los años cincuenta. Hopper detectó la necesidad de usar un lenguaje de programación amigable para el programador, un lenguaje de alto nivel parecido al inglés que permitiera lidiar con los errores y el tedio de programar en lenguaje binario. A partir de esto pensó en crear un compilador que sirviera como programa intermediario entre las instrucciones escritas y el código binario. A inicios de los años cincuenta, Hopper creó *Flow-Matic*, considerado el primer compilador de datos en inglés.

El live coding sonoro opera en un constante ir y venir entre la escritura de código fuente realizada por el live coder, su traducción y ejecución por parte de la computadora y la acción que resulta. Una presentación de live coding comienza cuando el live coder escribe las primeras líneas de código y luego las compila para que la computadora traduzca y ejecute el código en la acción de generar sonido; entonces, el live coder, con base en los sonidos que escucha, continúa escribiendo o modificando el código fuente. De esta manera, el *sonido emergente*[35] se actualiza cada vez que el live coder indica a la computadora que compile o interprete los cambios que ha realizando en el código fuente. El proceso de live coding se efectúa en una secuencia cíclica que comprende escritura, traducción, ejecución y acción del código[36]. Esta secuencia se mantiene durante la presentación de live coding en un bucle de generación y actualización del sonido con base en la escritura de código y las

35 Por sonido emergente me voy a referir en este trabajo al sonido que resulta cuando el código es ejecutado; un sonido que se desenvuelve en el tiempo y que cambia con relación al código que el live coder actualiza cada vez que realiza una modificación de éste.

36 Esto se puede observar en el siguiente video <https://youtu.be/tHhdzMOI7Sk> en el que el músico electrónico Daniele Filaretti reescribe código de manera alternada en dos programas —TidalCycles y SuperCollider— para modificar el sonido emergente. Se pueden escuchar una serie de cambios de la estructura musical que está sonando y además se pueden ver ciertos momentos en que un fragmento del código se ilumina. En esos momentos el live coder realiza una acción que consiste en pulsar una combinación de teclas cada vez que termina de escribir o reescribir código, es el punto en el que ordena a la computadora traducir el código fuente en código ejecutable que realiza o actualiza la acción. Como se observa, la música va cambiando en una secuencia de acciones que se repiten durante el video: escribir código e interpretarlo.

decisiones que toma el live coder para modificarlo a partir de lo que escucha. El live coder se ve envuelto en distintas etapas temporales que determinan su relación con el código: una etapa en la que escribe código fuente; una en la que le indica a la computadora que traduzca el código escrito; y una en la que la computadora ejecuta el código máquina que resulta de la traducción para producir sonido.

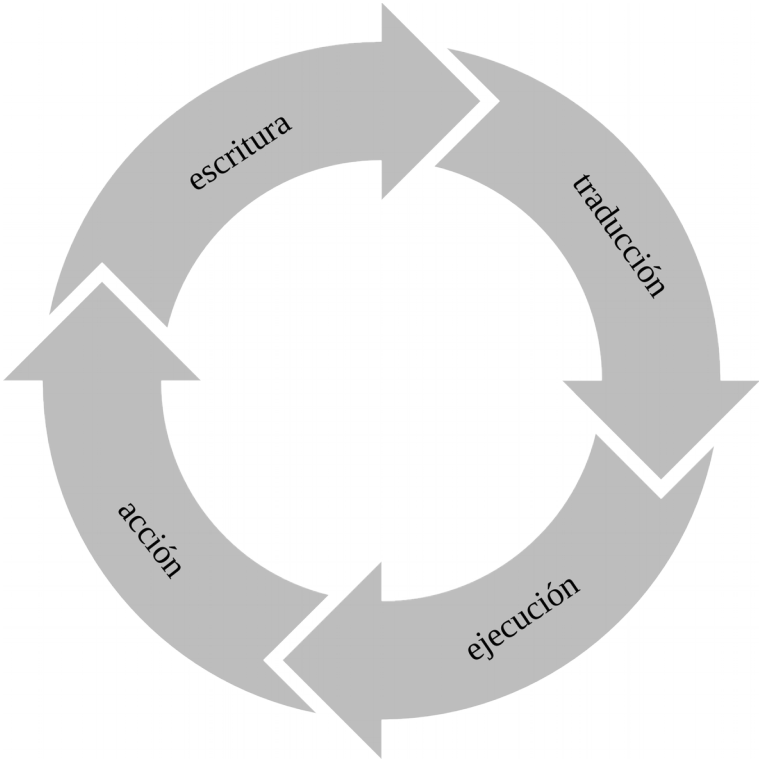


Figura 3. Ciclo del live coding

Sarah Bell (s.f) explica que cuando la escritura de código es simple y rápida hay una "retroalimentación creativa"[37] (párr. 5) entre la percepción de la música y el programa que se escribe. La escritura sencilla de código es una característica que los programas dedicados al live coding contemplan en su diseño, con tal fin utilizan lenguajes de programación de alto nivel, lo que facilita formular instrucciones para la computadora en

37 Creative feedback.

un tiempo reducido, como la situación de un concierto. De esta forma se logra una mayor conexión entre lo que se va escribiendo y lo que va sonando. Bell (s.f.) explica que la retroalimentación "es la base de la relación entre músico, código y audiencia, así que el lenguaje de programación necesita actuar rápido"[38] (ibíd.). En esta cita la autora se refiere a poder escuchar de manera inmediata los cambios del sonido emergente causados por las modificaciones y actualizaciones que se realizan en el código escrito durante el tiempo de concierto. Al respecto, Andrew Brown (2016) comenta que un componente importante de las prácticas de música algorítmica es la velocidad de retroalimentación o "el intervalo entre la música escrita como código y la escucha del resultado"[39] (p. 181).

Como se menciona, la forma rápida en la que actúa un programa en el live coding se debe al uso de un lenguaje de programación de alto nivel y una escritura reducida de código más que a la velocidad al teclear. En este sentido, Rohrhuber y De Campo (2009) proponen una escritura de código compacta en el momento de realizar live coding, pues mencionan que los algoritmos complejos[40] generan ambigüedad entre automatismo e intervención sobre el código, los autores se refieren al código que contiene una serie de instrucciones complejas que dejan poco espacio para reescribirlo o modificarlo en vivo. Un código complejo crea la sensación de que los cambios y la reescritura realizada por el live coder, sobre el código, no se reflejan en el sonido emergente. Para estos autores, el live coding dista de demostrar una escritura compleja y estructurada, pues el reto intelectual para live coders y audiencia es entender la causalidad de la modificación del código que se refleja en el sonido emergente cuando éste es modificado varias veces durante un concierto. La aproximación a una escritura basada en códigos pequeños y fáciles de modificar,

38 "It is the basis of the relationship between the musician, the code and the audience, so the programming language needs to work fast".

39 "the interval between notating music as code and hearing the result".

40 Los autores mencionan que el sonido algorítmico viene de una formalización y que en una improvisación es interesante escuchar esa formalización, sin embargo, dicen que aunque un algoritmo de comportamiento muy complejo suena bastante convincente, éste no sería la mejor opción para crear una conversación con el código pues es difícil entender el proceso de intervención sobre él.

posibilita una intervención continua sobre el código cuando responde rápido a los sonidos que produce la computadora; esta forma de interacción entre live coder y código fuente es lo que Rorhrhuber y De Campo (2009) llaman “programación conversacional”.

En este sentido, la práctica de live coding puede verse como una conversación entre el live coder y la computadora a través del código que escribe el primero y el sonido que regresa la segunda. Este resultado puede parecer incierto frente al proceso de escritura de código fuente, pues aunque éste contiene la intención de algo que queremos que suceda, no siempre llega a producirse cuando se ejecuta. Bajo tal circunstancia se toman decisiones durante un concierto para modificar el código que describe las acciones del sonido emergente. En la relación de lo que se escribe y el sonido que resulta se encuentra el diálogo entre live coder y computadora dentro de la improvisación con código.

Una forma de interactuar rápido con el código —no por teclear rápido— es organizar el código en patrones[41]. Un patrón recorre el código de inicio a fin de manera cíclica. Cada vez que modificamos el código de un patrón, el recorrido de la lectura es distinta, y eso, en el momento de ejecutarlo, cambiará el sonido emergente. La música creada con base en patrones consiste en organizar parámetros del sonido sobre el tiempo como la duración, la altura o el volumen de una nota musical o una muestra de sonido. Esto permite formar ritmos, melodías, armonías, texturas, densidades y estructuras que se desarrollan sobre el tiempo.

```
Pdef(\uno, Pbind(\dur, Pseq([0.5, 1], inf),\vol, 0.5, \midinote,
Pseq([60, 64, 65, 67, 72], inf))).play
```

Figura 4. Patrón en el que está programada la duración, el volumen y altura de una nota musical

41 Ron Kuivila (en Collins et al., 2011) define un patrón o *pattern* en SuperCollider como “representaciones abstractas de secuencias” (p. 189).

La figura 4 muestra código fuente escrito como patrón. Este fragmento contiene la instrucción para actuar sobre tres parámetros de una nota musical: la duración expresada como `\dur`, el volumen expresado como `\vol` y la altura de la nota musical expresada como `\midinote`.

Como se menciona al inicio del capítulo, lo que escribe y modifica el live coder, por un lado, es la fuente sonora, por el otro, su comportamiento para producir sonido; una combinación en la que se construye la fuente sonora mientras se activa para sonar. La estrategia más común es escribir primero la fuente sonora —en algunos casos se trabaja con muestras de sonido—, y en el momento de realizar live coding escribir y modificar su comportamiento mediante instrucciones contenidas en un patrón. A veces la misma fuente o instrumento contiene su comportamiento, como una cajita musical; ambos están escritos con el mismo lenguaje y en algunas ocasiones, contenidos en el mismo documento de texto.

Por último, durante una presentación de live coding existe una tensión entre el código que suena y el código que falla, entre el tiempo de programación y el tiempo que pasa para que el live coder tome una nueva decisión y modifique lo que suena, lo escriba y lo actualice. En este proceso, el código escrito puede fallar en el momento de su traducción por algún error en la sintaxis. Errar en la escritura de código es frecuente en las presentaciones de live coding; contrario a considerarse un defecto muestra que la práctica de escribir código se realiza en vivo; el error es parte del proceso.

1.4 Improvisación en el live coding

En los últimos catorce años el live coding se ha establecido como una práctica musical con énfasis en la improvisación. Como menciona McLean (2011), el término live coding “se utiliza con mayor frecuencia en el contexto del performance improvisado de música y

animación de video”[42] (p. 129). El presente trabajo comparte, en acuerdo con otros autores (McLean, 2011; Rohrhuber y De Campo 2009; Wilson et al. 2014), que el live coding es improvisación, y se acerca al tema desde el pensamiento de la improvisación libre descrito por algunos autores e improvisadores (Alonso, 2008; Bailey, 2010; Berenguer, 2007; Matthews, 2012; Moraes Costa, 2015). Con esto no sugiero a prior que el live coding es una práctica de improvisación libre, más bien encuentro una conexión entre ambas prácticas a través de sus discursos. Esta conexión sirve para argumentar más adelante la cualidad improvisatoria del live coding como práctica artística en red.

Según Wade Matthews (2012), la *improvisación* es una práctica de creación musical en la que “el creador comparte su proceso creativo con el público *en el acto*, está creando la música en ese preciso instante y lugar” (p. 27). Para Derek Bailey (2010), la música improvisada libremente, como él la llama, presenta un identidad confusa y es difícil de etiquetar. El autor dice que:

> Su principal característica es la diversidad. No se adhiere a ningún estilo o idioma. No tiene que conseguir ninguna clase de sonido particular. Las características de la música libremente improvisada se establecen solo a partir de la identidad musical de las personas que la tocan. (p. 164)

Pero ¿cómo se relaciona el live coding con la improvisación? A partir de lo antes expuesto, lo que caracteriza al live coding como práctica de improvisación es el acto de escribir y modificar código fuente en vivo para obtener un resultado sonoro en el momento, casi siempre este proceso es mostrado al público junto al resultado musical. Estas características ponen al live coding ante la posibilidad de considerarse una práctica de improvisación. En este sentido, el live coding se expresa como improvisación pues desde su significado denota una acción en proceso —codificar en vivo.

42 “live coding is most often used in the context of improvised performance of music and video animation”.

A lo largo de este texto, así como en la teoría sobre live coding, la palabra proceso aparece referida en distintas ocasiones para resaltar su carácter de improvisación (Collins et al., 2003; McLean, 2011; Rohrhuber y De Campo, 2009; Ward et al., 2004; Wilson et al., 2014). Pero no solo en la definición de live coding, también en la definición que varios autores hacen sobre la improvisación (Berenguer, 2007; Matthews, 2012; Moraes, 2015).

A partir de lo anterior, si se retoma la definición de Ward et al. (2004) que define al live coding como "la actividad de escribir (partes de) un programa mientras está corriendo" (véase §1) y se compara con lo que José Manuel Berenguer dice de la improvisación: "algo que recorre todo el proceso en el que la música se va creando" (2007, p. 42), se observa que ambas definiciones se refieren a actividades que están en el transcurso de creación —de un programa en el primer caso, de música en el segundo— y se enfatiza el recorrido sobre el punto de llegada. La idea de proceso, tanto en el live coding como en la improvisación, suele contraponerse, por los autores, a la idea de objeto terminado. Al respecto, Moreaes Costa (2015) dice que "el objetivo de la improvisación libre es generar procesos creativos y no obras acabadas"[43] (p. 120). Mientras que McLean (2011) comenta que el live coding es música por computadora improvisada que se enfoca en un proceso de creación más que en obtener un resultado; lo que el autor expresa de la siguiente manera:

> En el live coding, el performance es el *proceso* de desarrollo de software más que el resultado. La obra no es generada por un programa terminado, sino durante su recorrido de desarrollo que parte de la nada hacia un algoritmo complejo, generando continuamente, a lo largo del camino, formas musicales o visuales cambiantes[44]. (p. 130)

43 "O objetivo da livre improvisação é gerar processos criativos e não obras acabadas".

44 "In live coding the performance is the *process* of software development, rather than its outcome. The work is not generated by a finished program, but through its journey of development from nothing to a complex algorithm, generating continuously changing musical or visual form along the way".

La improvisación y el live coding no solo se definen en términos del proceso, involucran otros aspectos como la interacción[45]. En el live coding se puede mencionar la interacción que ocurre entre el live coder, su código y lo que produce de dos formas: una visual en la que el código textual se escribe con el teclado y se lee en la pantalla de la computadora y otra auditiva en la que el sonido emergente suena en las bocinas, ya sea de la computadora o de un sistema de sonido.

Para lograr una mejor interacción con el código Rorhuber y De Campo (2009) proponen sistemas de live coding que ayuden a los improvisadores a recordar y entender todo el estado del sistema y lo que puede causar. Por su parte, Matthews (2012) comenta sobre la sencillez en la improvisación libre: “Una forma compleja obliga al músico a estar demasiado pendiente de la estructura, reduciendo su capacidad de interactuar libremente con sus colegas. En cambio, un improvisador experimentado puede navegar por una forma sencilla con muy poco esfuerzo” (p. 44). Lo anterior se puede entender en el live coding a partir de la escritura de estructuras de código que permitan una retroalimentación constante entre lo que se escribe y lo que se escucha.

Para Wilson, Lorway, Coull, Vasilakos y Moyers (2014) el live coding es un tipo de improvisación dentro de la música con computadora o la improvisación se asume como aspecto implícito del live coding. Wilson et al. destacan las implicaciones de la improvisación a partir de utilizar el código fuente como interfaz que permite intervenir y reconfigurar de manera radical los sistemas musicales mientras corren, aunque detectan que hay una carencia de agilidad y destreza; y argumentan que así como sucede en la “improvisación post-free jazz”[46] (p. 54), el live coding presenta un problema en el momento de crear articulaciones formales, y que mucho del live coding musical, de igual manera que

45 El diccionario de la RAE define *interacción* como “Acción que se ejerce recíprocamente entre dos o más objetos, personas, agentes, fuerzas, funciones, etc.” (s.f.). Mientras que Herrera (2013) dice que “la interacción es un fenómeno de la comunicación que experimentamos de forma cotidiana en las relaciones interpersonales” (p. 14), el autor agrega que ésta sucede a través de un canal de comunicación en el que hay un “intercambio de mensajes de cualquier tipo” (ibíd.).

46 “post-free jazz improvisation”

la improvisación libre, “consiste en un flujo gradual en el que se añaden y sustraen capas, con algunas modificaciones en medio”[47] (ibíd).

Por otro lado, Berenguer (2007) advierte que la improvisación no surge de la nada, que este proceso comienza fuera de la escena y que el improvisador dedica tiempo para trabajar e investigar sus materiales antes de experimentar con ellos en una improvisación. En este aspecto, al improvisar con código recurrimos a una serie de conocimientos especializados sobre programación que sabemos el tipo de sonido que generan. Crear en el momento cuando se realiza live coding no significa partir de la nada, siempre hay referentes, la creación en el momento, a partir de la escritura de código, es posible por un conocimiento especializado del lenguaje de programación que alimenta la memoria con lo que produce el código. Esto contradice lo que comenta Moraes Costa (2015), quien plantea que en la improvisación libre “no existen materiales pre-elaborados”[48] (p. 125), más bien se construye a partir de las intervenciones de los improvisadores en un proceso interactivo y empírico que “va dando forma al flujo sonoro”[49] (ibíd).

Respecto a lo anterior, Rohrhuber, De Campo, Wieser, Van Kampen, Ho y Hölzl (2007) comentan que un punto central del live coding es el desplazamiento del pensamiento de un lugar privado a un lugar público. Con esto, se refieren a pensar el código que se escribe durante un concierto frente a una audiencia y lo llaman “razonamiento público”[50] (párr. 7). El razonamiento público, vinculado a la idea de creación en el momento de la improvisación, no comienza en el escenario, pues como aclaran los autores, comienza en un espacio privado, ya sea el estudio del músico o el lugar de ensayo.

Por otra parte, improvisar con código no es mostrar un proceso virtuoso al escribir rápido, es más bien arriesgarse y explorar el sonido a partir de la escritura y modificación de código. Esto quizá implica una forma poco ortodoxa de programar y un proceso

47 “music consists of a gradual flow in which layers are added and subtracted, usually with some modifications in the interim”.

48 “não existem materiais pré-elaborados”.

49 “vão dando forma ao fluxo sonoro”.

50 “public reasoning”.

accidentado de escritura en el que se exploran los recursos del lenguaje de programación. Una exploración del sonido a través de la escritura de código, en los términos del live coding es, de manera paradójica, una situación controlada, pues para que el código ejecute una acción debe estar libre de errores de sintaxis, de otra manera no habrá sonido. El live coding presenta esta discrepancia; por un lado, es posible explorar formas inusuales de escritura de código, por otro, hay que controlar lo que se escribe. Si bien, escribir código nos permite expresar ideas libremente, su escritura depende de reglas para evitar errores.

Por último, la improvisación en el live coding no radica solo en la escritura de código, también en las acciones que produce en cuanto proceso dinámico para generar sonido. El live coding como práctica de improvisación involucra proceso, interacción, expectativa, incertidumbre, conocimiento y memoria. En palabras de Mattin (en Mattin y Iles, 2009), la improvisación libre es un terreno de inseguridad que alienta a una mayor improvisación y cuestiona el conocimiento personal y lo que se hace con él, creando un terreno inestable que rompe con restricciones previas.

Sobre este capítulo se puede concluir que el live coding es un modo de hacer música a partir de la escritura y modificación de código fuente en una secuencia cíclica en la que el live coder establece una relación textual con el sonido para actualizar su estado varias veces durante una presentación. La forma de traducción que mejor se adapta es aquella realizada por los lenguajes interpretados y de alto nivel, estos permiten al live coder compilar líneas de código por separado y le facilitan una escritura sencilla que permite una interacción rápida con el sonido. Las características técnicas descritas posibilitan al live coder realizar una práctica de creación en el momento e indagar en dinámicas propias de la improvisación. En el siguiente capítulo se discuten una serie de condiciones que tienen su origen en los años setenta, las cuales repercuten de manera posterior en el live coding y que sirven para argumentar una asociación entre la música en red, la improvisación libre y el movimiento hacker.

2. Live coding en red: práctica musical interconectada

El capítulo anterior describe la práctica musical de *live coding* y su relación con la *improvisación* a partir de la escritura y modificación de *código fuente* en el momento para generar sonido. Esta actividad es definida desde la interacción entre el *live coder,* o practicante de live coding, y su computadora a través de la escritura en un lenguaje de programación. Pero ¿qué sucede cuando varios live coders interactúan para realizar live coding?, ¿cómo se configura esta práctica en el momento de crear música a través de una red de computadoras?

Este capítulo establece el contexto tecnológico que enmarca la práctica de live coding en red, con base en la definición de *red de computadoras* (Barceló, Íñigo, Martí, Peig y Perramon, 2004; Tanenbaum y Wetherall, 2012). Así mismo, desarrolla la categoría de *live coding en red* a partir del concepto *networking* de Tatiana Bazzichelli (2008), el cual apunta a la red de relaciones sociales que establecen los integrantes de un grupo en el momento que realizan una práctica colectiva. A partir de ello, el capítulo traza una línea entre el live coding y las prácticas de networking, a través de una serie de asociaciones que atraviesan la música en red, la improvisación colectiva, las ideas de transformación (Sorensen, 2007) e influencia mutua (Borgo, 2006) y las acciones de compartir e intercambiar implícitas en las funciones de las redes de computadoras y expresados como valores en la ética hacker y los movimientos de software libre y open source.

2.1 Red de computadoras: la posibilidad de intercambiar y compartir

Barceló et al. (2004) atribuyen el surgimiento de las redes de computadoras a una necesidad de establecer una comunicación entre éstas para compartir información. Según los autores,

el origen se encuentra en la red telefónica, sistema cuya función es la comunicación a distancia. Mencionan que desde 1878, cuando Alexander Graham Bell mostró por primera vez cómo realizar conversaciones entre dos aparatos remotos unidos por un cable, hasta la década de los años setenta —un periodo de aproximadamente 100 años— se desarrolló una compleja infraestructura de redes telefónicas a nivel mundial para la comunicación internacional que fue aprovechada ya desde los años sesenta para conectar las primeras computadoras comerciales. Según los autores, estas computadoras, al ser muy costosas, fueron diseñadas como sistemas multiusuario que permitían realizar distintas tareas de manera simultánea en una sola computadora. El acceso remoto a dichas computadoras se realizaba a través de las redes telefónicas con la ayuda de un aparato llamado módem[51].

Por otro lado, O'Regan (2010), menciona que en el año 1965 se estableció la primer red de área amplia para conectar dos computadoras localizadas en distintos puntos geográficos, una en el Instituto Tecnológico de Massachusetts o MIT y otra en Santa Mónica. Según el autor, esta conexión demostraba que una línea de teléfono era viable para la trasmisión de datos a la distancia. Además, cuenta que a inicio de los años sesenta, la Agencia de Proyectos de Investigación Avanzada o ARPA[52], trabajaba en el desarrollo de un sistema de comunicación de mensajes entre computadoras llamado “conmutación de paquetes”[53] (p. 64), lo que más adelante permitiría la transmisión de datos en las redes de computadoras.

Tanenbaum y Wetherall (2012) dicen que el modelo de computadora central que realiza todas las tareas —el de múltiples usuarios conectados a una sola computadora— ha sido cambiado por el de muchas computadoras separadas e interconectadas que realizan el trabajo; y que estos sistemas son los que se conocen como *redes de computadoras*. Los

51 Según Bareceló et al., el módem se creó para adaptar lenguaje binario a una red telefónica analógica, lo cual permitió transferir datos en una infraestructura diseñada para transmitir sonido dentro de un rango de frecuencia específico.

52 ARPA es creada, según O'Regan (2010), por el Departamento de Defensa de Estados Unidos a finales de 1950, con el objetivo de desarrollar nuevas tecnologías para la milicia de ese país.

53 “Packet switching”.

autores las definen como un “conjunto de computadoras autónomas interconectadas mediante una sola tecnología” (p. 2); además agregan que “dos computadoras están interconectadas si pueden intercambiar información” (ibíd.). Esta idea es clave para entender el discurso sobre el intercambio que posibilitan las redes; intercambio que sucede a partir de la posibilidad técnica, el diseño y la configuración de las computadoras conectadas en red y que permite a varios individuos comunicarse, a través de sus computadoras, para establecer dinámicas de intercambio y compartición de información en la que hay ideas y conocimientos implícitos.

Un objetivo en la creación de redes de computadoras es la comunicación a la distancia (Barceló et al, 2004; O'Regan, 2010), lo cual tiene una conexión con el uso de redes en un contexto artístico. Chandler y Neumark (2005) plantean que el arte, el activismo y los medios se han reconfigurado entre ellos a la distancia; los autores plantean que esto sucede en la segunda mitad del siglo XX cuando los artistas comienzan a utilizar medios de comunicación como medios de arte, por ejemplo, el radio, el servicio de correos o los satélites. Bazzichelli (2008), plantea que el arte en redes comienza fuera de la informática y las telecomunicaciones, y establece sus inicios en el movimiento artístico del mail art[54]. Al respecto, John Held (en Chandler y Neumark, 2005) menciona que los artistas del mail art establecieron contacto a través de la red internacional del servicio postal que sirvió como un “sistema abierto de comunicación”[55] (p. 89) que disolvía las barreras impuestas por las fronteras y les permitía distribuir arte e ideas cargadas de un lenguaje e ideología política. Posteriormente, en una etapa del Internet, Juan Martin Prada (2012) dice que el arte de la red junta comunicación y comunidad, así como distribución y accesibilidad

54 Bazzichelli (2008) define el mail art como “una red de relaciones construida a través del circuito postal” [Mail art is a network of relationships constructed through the postal circuit] (p. 37). Según la autora, consiste en el envío y recepción de cartas, postales o cualquier cosa a nivel mundial que establece conexiones entre individuos con el solo interés de comunicarse. Con poco reconocimiento en el mundo del arte, esta práctica está abierta para todos y Bazzichelli la considera la madre del networking. La autora agrega que el nombre original de esta práctica fue “*Eternal Network*” (ibíd.), acuñado por Robert Filliou a inicios de los años cincuenta.

55 “open system of communication”.

en el contexto de la Web 2.0, donde el valor principal es la compartición e intercambio colectivo de datos promovido por el deseo de llevarlo a cabo. Esto tiene que ver con la posibilidad técnica que ofrece la interconexión de computadoras, pero también con lo que expresa el discurso del movimiento open source, el cual se origina a raíz del pensamiento del software libre.

Según DiBona, Okcman y Stone (1999), el software libre fomenta la compartición y distribución del código fuente con el que está hecho un programa. Richard Stallman[56] equipara el código fuente con el conocimiento científico por lo que piensa que debe distribuirse de manera libre y que nadie debe pagar por software. El open source deriva de este pensamiento aunque trata de conciliar el rechazo hacia el software libre, esto ha implicado, bajo la iniciativa de Eric S. Raymond[57], cambiar la postura hacia el software libre y crear un término más atractivo para la industria y quienes ven con desconfianza el discurso libre en torno al software y su código. Lo anterior sucede bajo el término open source, el cual, desde el año 1997, opera con lineamientos que permiten juntar el software libre con el software propietario bajo un principio de compartir. Este esquema, a diferencia del software libre, genera un modelo económico. Los autores plantean que el open source, al compartirse, puede ser replicado de igual manera que el método científico lo hace con el conocimiento. Programar sin buscar una compensación fomenta la búsqueda de reputación a partir de una cultura del regalo donde el valor del trabajo recae en compartirlo con otros. Lo que se observa en ambos discursos —software libre y open source— es el énfasis en el código fuente, en su compartición y distribución, aunque con objetivos distintos[58]. Este pensamiento permea al live coding a través del desarrollo de software que se adscribe a los discursos y principios antes mencionados, a partir de lo que se crea un campo de la música por computadora que opera con el uso de software libre y open source[59].

Steven Levy, en su libro *Hackers. Heroes of the Computer Revolution* escrito en

56 Uno de los principales promotores de software libre quien en 1984 creo el proyecto GNU para impulsar su desarrollo y difusión a través de *Free Software Foundation*.

57 Hacker promotor del movimiento open source, escribió el ensayo *The Cathedral and the Baazar* y es co-fundador de *Open Source Inititive* (DiBonna et al. 1999).

1984, define por primera vez el término “ética hacker” en el que describe al *hacker* como un sujeto que conoce el funcionamiento de las computadoras y la programación de estas con una filosofía de compartir, abrir, descentralizar y explorar las computadoras usándolas con el objetivo de mejoralas y mejorar el mundo, para lo que, la información necesaria para conocer su funcionamiento debe ser libre. Por su parte, Pekka Himanen (2001) define a los hackers, a partir del documento en línea llamado *jargon file*[60], como programadores entusiastas que creen que compartir información, experiencia, software libre y recursos de computación es un bien y deber ético. El autor sitúa la figura del hacker en un contexto que tiene como base tecnológica de la sociedad al Internet, la computadora personal y el software.

Por último, el live coding no se puede definir solo en términos de la tecnología que utiliza, tendría que ser, a su vez, en términos de la interacción y las relaciones que se generan entre los integrantes del grupo, puesto que una red no solo conecta computadoras, también individuos. Rohrhuber (2007) lo expresa al concluir que el uso de redes en la música plantea dos cuestiones: cómo transmitir y cómo formar redes de relaciones. Al respecto, Thacker (en Galloway, 2004) menciona que las redes no son metáforas de interconectividad más bien “tecnologías materiales, sitios de prácticas variables, acciones y movimientos” (p. xiii). Esto se conecta con la práctica de música en red a través de lo planteado por Brown y Bischoff (en Chandler y Neumark, 2005), quienes dicen que “la red es lugar y origen para un nuevo tipo de música” (p. 373), la cual consideran una forma

58 Richard Stallman menciona, en la conferencia *Software Libre, Diseños Físicos Libres* (Fábrica Digital El Rule, Ciudad de México, 31 de octubre de 2017), que una diferencia entre los objetivos del software libre y el open source es que el primero apela a valores éticos a través del activismo y la demanda de libertad, mientras que el segundo a valores prácticos a través de reforzar y mejorar la calidad. Stallman hace énfasis en tener acceso al código fuente del software debido a que es la parte legible que le permite al usuario entenderlo y modificarlo.

59 Por ejemplo, al revisar los sitios web y el código fuente de SuperCollider, TidalCycles y ChucK se observa que son distribuidos bajo una licencia *GNU Public license.* Mientras que Sonic Pi se declara como un proyecto open source desarrollado bajo una licencia MIT.

60 Glosario de argot hacker <http://www.catb.org/~esr/jargon/>

tecnológica que puede ser manipulada colaborativamente, lo que permite trascender las barreras físicas de los individuos.

2.2 Música en red

La idea de interconectar computadoras para crear música no es nueva, al live coding lo antecede la corriente conocida como *network music* que tiene su origen en Estados Unidos a mediados de los años setenta dentro de la escena de música experimental del Área de la Bahía de San Francisco. Brown y Bischoff (2002) cuentan que varios compositores radicados en esta zona comenzaron a utilizar la microcomputadora KIM-1[61] para componer música electrónica programada en el lenguaje de máquina 6502[62] bajo la iniciativa de Jim Horton. Entre estos compositores se encontraba Rich Gold y el propio John Bischoff quienes comenzaron a colaborar con Horton en la creación de piezas musicales con las microcomputadoras conectadas a través de su puerto serial. Más tarde formarían la primer banda con computadoras interconectadas conocida como The League of Automatic Music Composers.

Según Rohrhuber (2007), el campo de network music "cubre un amplio rango, desde ambientes colaborativos de composición hasta instalaciones sonoras y ensambles de música improvisada"[63] (p. 140), en el que la composición y la improvisación se llevan a cabo con el uso de redes locales o a través de Internet. Así mismo, Rohrhuber comenta que

61 Según Brown y Bischoff (2002), la microcomputadora KIM-1 —llamada así por su tamaño pequeño comparado con las grandes computadoras centrales— fue una de las primeras computadoras de consumo personal en el mercado. Los autores refieren que por primera vez fue posible utilizar un computadora de manera personal sin depender de las computadoras de las instituciones.

62 Según el sitio <http://www.6502.org/trainers/> el lenguaje ensamblador o lenguaje de máquina 6502 era utilizado por las micricomputadoras o *trainers* MOS KIM-1 y Rockwell AIM-65.

63 "It covers a broad range from collaborative composition environments to sound installations and improvised music ensembles".

la música en red es un tipo de arte que se basa en técnicas propias de la telecomunicación.

La aproximación al trabajo con redes de computadoras en la música se ha prestado para explorar una forma de creación grupal en la que compartir e intercambiar datos, información, conocimientos e ideas tiene un papel central. La idea de interconectar computadoras para hacer música, según Brown y Bischoff (2002) —pioneros de la música en red e integrantes del grupo *The Hub*— surge en un contexto de ideales y expectativas que giraban en torno a una utopía política con base en "una nueva sociedad construida en el acceso libre y abierto de la información" (párr. 3).

El live coding practicado en red es un caso de network music que refleja el deseo de crear música en grupo, de compartir una tecnología y de intercambiar conocimiento relativo a la práctica. Tatiana Bazzichelli (2008) ha investigado el concepto *networking* a partir de prácticas en red que se desarrollan bajo el principio de la colaboración entre individuos conectados por diferentes infraestructuras, desde el correo postal hasta las redes de computadoras. El siguiente apartado expone este concepto para después trazar una línea entre las prácticas de networking y el live coding en red.

2.3 Networking

Networking es hacer red, asienta Tatiana Bazzichelli (2008) en su libro *Networking: The Net as Artwork;* a partir de esta frase la autora expresa que "hacer red significa crear redes de relaciones, para compartir experiencias e ideas en un contexto de intercambio comunicativo, y en una experimentación artística en la que el emisor y el receptor, el artista y el público, actúan en el mismo plano"[64] (p. 18). Estas relaciones, según la autora, están en proceso y se forman bajo dinámicas abiertas; además, agrega que hacer red "también

64 To network means to create relationship networks, in order to share experiences and ideas in the context of a communicative exchange, and an artistic experimentation in which the sender, and the receiver, the artist and the public, act on the same plane.

significa crear contextos en los que la gente pueda sentirse libre para comunicar y crear artísticamente de manera 'horizontal'"[65] (p. 26). La intención de la autora, al estudiar el concepto de networking, es ahondar en actividades que articulan la construcción de conexiones y redes de relaciones entre individuos que proponen un uso creativo, compartido y consciente de la tecnología.

Para abordar el concepto de networking la autora indaga sobre el origen del término y realiza su reconstrucción a partir del análisis de prácticas y movimientos artísticos en los que la participación, interacción y colaboración son centrales, tal es el caso de Fluxus[66], mail art, net.art[67]; prácticas tecnológicas como los Bulletin Board System o BBS[68]; el movimiento DIY[69], el movimiento activista hacker[70] y el género distópico

65 It also means to create contexts in which people can feel free to communicate and to create artistically in a "horizontal" manner.

66 Peter Frank (en Crawford, 2008) define Fluxus como un movimiento y un grupo de artistas de vanguardia que trabajaron juntos desde finales de los años cincuenta. El autor menciona que se le atribuye a George Maciunas el término Fluxus; término que le dio identidad a este colectivo de artistas que operó como una red abierta e informal con una esencia política y colaborativa.

67 Juan Martín Prada dice que el *net art* comienza a mediados de los noventa cuando "muchos artistas comenzaron a hacer de Internet el ámbito de investigación principal de sus obras y su contexto de actuación específico, considerándolo como un medio o espacio en el que intervenir y sobre el que reflexionar creativa y críticamente" (2012, p. 9). Prada hace una acotación para el término *net.art* (con punto) en la que especifica que se refiere a los primeros artistas que trabajaron en los los noventa y que se les identifica como *net.art group* (Vuk Ćosić —quien acuñó el término—, Alexei Shulgin, Jordi.org, Heath Bunting y Oliana Lialina).

68 Los BBS son sistemas de transferencia de datos entre computadoras que operan vía telefónica y que algunos programadores han utilizado para intercambiar software, información y noticias antes de que existiera Internet.

69 DIY son las siglas para *Do It Yourself* (Hazlo Tú Mismo). Bazzichelli (2013) define el término como una "'actitud' de construir, ensamblar, hacer y crear de manera independiente" ["attitude" to build up, to assemble, to make and to create independently] (p. 71).

70 La autora hace referencia al término Hacktivismo, una práctica de activismo tecnológico en la que están involucrados individuos con una postura política.

Cyberpunk[71]. A las prácticas colectivas enmarcadas bajo estas corrientes y movimientos la autora las llama “prácticas de networking” (Bazzichelli, 2008; 2013) y, aunque se desarrollan en contextos y periodos distintos a lo largo de la segunda mitad del siglo XX, tienen en común la conexión que generan diferentes individuos en el momento que realizan una actividad colectiva. Con base en dichas prácticas la autora define el concepto de networking de la siguiente manera:

> Estas prácticas escasas han mostrado que el arte en red no se determina principalmente de manera tecnológica, sino que está basado en la creación de plataformas de compartición y en contextos que permiten un intercambio entre individuos y grupos. Esta perspectiva hace posible definir el concepto de networking de tres maneras: como la práctica de crear redes de relaciones; como estrategia cultural dirigida a compartir conocimiento y como un mapa de conexiones en progreso[72]. (Bazzichelli, 2013, p. 71)

La autora expone tres ideas centrales con el concepto de networking, una es que

71 Bazzichelli (2008) menciona que el Cyberpunk está ligado al género literario del mismo nombre desarrollado por William Gibson y Bruce Sterling que es reinterpretado en prácticas políticas y de lo cotidiano principalmente en Italia durante los años noventa dentro de los espacios llamados Centros Sociales (*Squatts* en Europa), que según la autora, han sido territorios de experimentación libertaria en una primer etapa utópica en la que se difundieron tecnologías como los módems y las computadoras.

72 These narrow practices have shown that networked art is not mainly technologically determined, but is based on the creation of sharing platforms and of contexts which permit exchange between individuals and groups. Such a perspective made it possible to define the concept of networking in three ways: as the practice of creating nets of relations; as a cultural strategy aimed at sharing knowledge, and as a map of connections in progress. La traducción de *narrow* en prácticas “escasas” se refiere a prácticas colectivas en red antes del 2000, las cuales, como menciona la autora, no eran tan comunes y sucedían en contextos contraculturales y *underground*. Esto es contrario a lo que el término significa después del 2000, el cual según Bazzichelli (2013), se emplea para nombrar estrategias de negocio y el modelo de interacción de la Web 2.0. La autora usa el término “social networking” para referirse a una actividad que sucede cotidianamente en la Web.

las prácticas colectivas mencionadas construyen redes de relaciones a través del intercambio y la compartición de conocimientos, ideas y experiencias en contextos que enmarcan la creación artística de grupos que comparten el uso de una tecnología o infraestructura. Por ejemplo, una red de computadoras en el caso de las plataformas de compartición de datos previos a Internet llamados *Bulletin Board Systems* o la red de correo postal en el caso del *mail art*.

La autora menciona que algunas de estas redes son creadas por artistas quienes adquieren el papel de "operador de red"[73] (p. 26) o *networker*[74]. En palabras de Bazzichelli (2008) "el arte de networking se basa en la figura del artista como creador de plataformas de compartición y de contextos para conectar e intercambiar"[75] (p. ibíd.). Según Bazzichelli, el artista adquiere un nuevo papel al convertirse en operador de redes colectivas dentro del arte del networking y a partir de una serie de asociaciones, conecta la figura del activista que experimenta usando tecnología en red con prácticas de la neovanguardia como Fluxus o mail art a través de la figura del networker.

La segunda idea que expone Bazzichelli (2013), para definir el concepto de networking, es la de estrategia cultural dirigida a compartir conocimiento. La autora menciona que intercambiar y compartir conocimiento para crear situaciones estéticas es común en prácticas artísticas realizadas por grupos pequeños en la última mitad del siglo XX. También menciona que la idea de intercambio y compartición a través de la participación colectiva en redes tecnológicas se ve expresada en los valores de la ética hacker, la cual comienza a difundirse a mediados de los años ochenta, que en parte se ve expresada en los movimientos de software libre y open source.

Así mismo, Bazzichelli plantea una tercer idea en la que define al networking

73 "network operator".

74 Bazzichelli (2008) atribuye al movimiento de mail art la creación de la figura del operador de red, el cual puede ser visto en la la obra de Vittore Baroni *Real Corrispondence 6* de 1981. Posteriormente, menciona la autora, se emplearía el término *networker* dentro del ámbito del arte por computadora.

75 "The art of networking is based on the figure of the artist as creator of sharing platforms and of contexts for connecting and exchanging".

como un mapa de conexiones en progreso; y que "[…] no se basa solo en objetos, ni en instrumentos digitales o analógicos, sino en las relaciones y procesos en progreso que se dan entre los individuos"[76] (Bazzichelli, 2008, p. 26). El resultado de estas prácticas, según la autora, no es un objeto o producto, es la práctica artística en progreso que a su vez crea una red como proceso abierto de creación colectiva; además, menciona, realizar una práctica colectiva crea una red de relaciones que a veces tiene más relevancia que el resultado mismo de la práctica. En este sentido, Bazzichelli (2008) propone que la red de relaciones puede considerarse una obra de arte.

A partir del concepto de networking, en las siguientes secciones se establece un diálogo con distintos términos, ideas y conceptos que trazan una línea hacia la práctica de live coding en red. Esta asociación se puede establecer ya que el live coding en red es una práctica de creación colectiva realizada con una tecnología de red que permite, entre otras cosas, compartir e intercambiar código y recursos, establecer una comunicación entre los participantes y trazar un mapa de la actividad a partir de las acciones que quedan inscritas en el código fuente que cada integrante escribe durante la práctica en red. Esta asociación, como se menciona, se desarrolla a través de un diálogo con la improvisación libre, la música en red y el movimiento hacker.

2.4 Live coding en red: una práctica de networking

Para definir el término *live coding en red* parto de la práctica artística que combina la escritura de código fuente en el momento, la actividad grupal y el uso de una red de computadoras para compartir, intercambiar, comunicar e interactuar en el momento de improvisar música de manera conjunta. Cabe mencionar que existen diferentes términos

76 […] it is not based on objects, nor solely on digital or analogical instruments, but on the relationships and processes in progress between individuals.

que describen esta práctica tales como *networked live coding*[77] (Lee y Essl, 2014; Rohrhuber et al., 2007; Wilson et al., 2014), *collaborative live coding*[78] (Lee y Essl, 2015), *networked ensemble performance*[79] (Roberts, Yerkes, Bazo, Wright y Kuchera-Morin, 2015) o *collective live coding*[80] (Ogborn, 2016). En este trabajo, sin embargo, considero pertinente construir el término a partir de una serie de asociaciones que me permitan conectar el live coding en red con las prácticas de networking. La razón por la que me interesa trazar esta conexión es para entender la práctica grupal de música que realizo desde una perspectiva técnica y social.

Cuando se realiza live coding en red, los eventos sonoros que genera cada integrante, mediante la escritura de código y su ejecución, se combinan para construir el sonido grupal. Este proceso se lleva a cabo a partir de una interacción grupal mediada por la interconexión y la escucha; en este proceso se construye el sonido en grupo en el que los integrantes del ensamble crean una red de relaciones. Al respecto, David Ogborn (2016) comenta que tanto la música como el software brindan una condición en la que se negocia "el despliegue de una red de relaciones en el tiempo"[81] (p. 18).

Para describir cómo se construye el sonido grupal en el live coding en red, me remito a dos ideas que pueden conectar con la noción de la creación de redes de relaciones

77 El live coding en red, según Rohrhuber et al. (2007), surge de la combinación de dos campos de la música por computadora: *live coding* y *network music*. Según los autores, en esta disciplina, el código se vuelve "un medio de comunicación y un pensamiento musical colectivo" [a means of communication and of collective musical thought] (párr. 8).

78 Lee y Essl (2015) describen live coding colaborativo a partir de tres aspectos que califican importantes: compartir código, entender el pensamiento del otro y el intercambio de ideas.

79 Roberts et al. (2015) utilizan el término *networked ensemble performance* para referirse a las presentaciones de live coding en red.

80 David Ogborn (2016) llama live coding colectivo a la práctica de live coding realizada por orquestas de laptop y ensambles similares. El autor argumenta que esta práctica es un punto de partida que va de la tradición de hacer música electrónica de manera solitaria en un estudio a nuevas formas de exploración de la co-presencia o intersubjetividad en los conciertos de live coding.

81 "the unfolding of a web of relationships in time".

del networking. La primera es la idea de transformación, descrita en el concepto de patrón de relaciones fluido de Estrid Sorensen (2007) y la segunda es la de inspiración mutua en la improvisación grupal descrita por David Borgo (2006).

En el artículo *The Time of Materiality*, Sorensen (2007) describe procesos de comunicación, interacción y relación entre individuos y objetos a partir del análisis de la actividad de un grupo de primaria en el salón de clase y en el laboratorio de cómputo de su escuela con el objetivo de explicar la materialidad de tres objetos —un pizarrón, una cama-desván y un entorno virtual— con lo que argumenta que la materialidad puede estudiarse en términos del espacio. Para este fin, Sorensen utiliza el concepto "patrones de relaciones"[82] (párr. 11) y define cada objeto a partir de uno de los siguientes: "patrón de relaciones regional"[83] (párr. 22), "patrón de relaciones red"[84] (párr. 31) y "patrón de relaciones fluido"[85] (párr. 35).

La idea de transformación que tomo de Sorensen, para construir la categoría de live coding en red, está descrita en el patrón de relaciones fluido, el cual hace referencia al trabajo del grupo de primaria para resolver tareas en un entorno virtual de computadoras. Este patrón sucede en forma de una secuencia en la que "*elementos individuales se unen uno después de otro*, sin seguir un plan fijo y sin estar dirigidos a un objetivo"[86] (párr. 34), donde "cada nuevo elemento influye en el *flujo* del procedimiento constituido por el ensamblaje de elementos"[87] (ibíd.). Sorensen menciona que las relaciones entre los

82 "patterns of relations". Sorensen desarrolló la noción *patrón de relaciones* para "describir la materialidad a través de patrones formados por las relaciones que una entidad particular, en concierto con otras entidades, establece para estas otras entidades particulares" [to describe materiality through the patterns formed by the relations a particular entity in concert with other entities establishes to these particular other entities] (2007, párr. 11).

83 "regional pattern of relations".

84 "network pattern of relations".

85 "fluid pattern of relations".

86 "[…] *single elements coming together one after the other*, neither following any set plan nor directed towards a goal".

87 "Each new element influenced the *flow* of the procedure constituted by the assemblage of elements".

elementos son inestables y están en transformación continua, son variables y por lo tanto no colapsan al cambio sino que dependen de incluir nuevos elementos.

Aunque Sorensen no se refiere a una actividad musical con la idea anterior, la autora describe un proceso de interacción parecido al que ocurre en la improvisación grupal, actividad que más allá de ser solo un intercambio de acciones, resulta en una "transformación continua" (Sorensen, 2007, párr. 36) de elementos sonoros y acciones que los integrantes de un grupo de improvisación realizan para dar forma al sonido grupal. La idea que quiero expresar al tomar este concepto es que los eventos sonoros y las acciones individuales que suceden en una improvisación de live coding en red no ocurren como una suma, más bien como una combinación que construye el sonido grupal emergente. Como se menciona antes, en esta acción se crea la red de relaciones al improvisar.

Al realizar live coding en red escuchamos, en primera instancia, el sonido que producen los demás y con base en éste tomamos decisiones para realizar modificaciones al código fuente. Esto, a su vez, ocurre por una influencia de las acciones y modificaciones que realizan los demás integrantes del grupo. David Borgo (2006), al hablar de improvisación libre, recupera la idea de "flujo de grupo"[88] de Keith Sawyer quien dice que al improvisar en conjunto los músicos logran "tocar cosas que no serían capaces de tocar solos o que no habrían explorado sin las inspiración del grupo"[89] (p. 3). Al igual que la idea de transformación expuesta por Sorensen en el patrón de relaciones fluido, la idea de inspiración mutua planteada por Borgo refuerza que el sonido emergente de la improvisación grupal es resultado de un proceso en el que la actividad de cada integrante influye en el desarrollo de la práctica colectiva.

Bajo estas dos ideas, se puede decir que los procesos en progreso de la red de relaciones, propuestos en el concepto de networking, se construyen en el momento de la improvisación, en una transformación continua de los eventos sonoros y en influencia

88 "Group flow".

89 […] to play things that they would not have been able to play alone or would not have explored without the inspiration of the group.

mutua entre los integrantes del ensamble. El código que circula en la red, como producto de una escritura individual, se ve afectado por la interacción con el grupo.

El live coding en red, al igual que la improvisación colectiva, es una práctica en la que se establecen relaciones entre los integrantes en el momento de la creación musical. En este sentido, Matthews (2012) expresa que "la libre improvisación no solamente es un acto de creación artística, sino también un acto *social*" (p. 61). El autor señala lo anterior, pues comenta que la improvisación colectiva se equipara a una "buena conversación" (ibíd.) cuando los interlocutores están dispuestos a dialogar. Al respecto, Rohrhuber et al. (2007) mencionan que en la práctica de live coding en red "el código no es solo la exhibición pública del razonamiento del intérprete, más bien se convierte en elemento constitutivo de una conversación entre los músicos"[90] (párr. 8). Al igual que en el comentario de Matthews, esta conversación apunta a la interacción que sucede durante una improvisación entre los integrantes de un grupo.

En este sentido, Chefa Alonso (2008) comenta que "la improvisación es una actividad colectiva; una de las finalidades es explorar la relación con los otros músicos" (p. 14). De esta manera puedo sugerir lo siguiente: si consideramos el live coding en red como un caso de improvisación colectiva podemos decir que es una práctica de networking pues improvisar en grupo crea una red de relaciones en la tarea de construir el sonido grupal. Además, el live coding en red tiene en común con la improvisación libre valores como compartir e intercambiar, y en ambos casos hay una cierta intención para que la práctica suceda en un contexto horizontal. Matthews (2012), al referirse a la improvisación libre en grupo dice lo siguiente:

90 […] code is not only a public display of performer's reasoning, but it becomes a constitutive element of conversation between the musicians.

> El modelo que subyace a la libre improvisación no inspira directamente el contenido sonoro; no es siquiera un modelo sonoro. Dicta, u ofrece, recursos de interacción entre los músicos. El contenido sonoro es el fruto de esa interacción, y además su vehículo. El sonido es, pues, el material con el cual los músicos negocian su relación con los demás, y pueden elegir materiales muy variados para entablar o vehiculizar tales relaciones. (p. 50)

Esta idea es importante pues refuerza el argumento que el live coding en red es una práctica de networking en tanto improvisación grupal en la que se establece una red de relaciones en la construcción del sonido grupal emergente. Alonso (2008), a su vez, menciona que la improvisación colectiva permite explorar las relaciones entre individuos al momento de improvisar:

> Esta disciplina es un laborioso proceso de exploración, no solo de las posibilidades del propio instrumento, para reflejar las necesidades expresivas de cada músico, sino también de las infinitas maneras de negociación social que se tienen que establecer entre los músicos que participan de esta experiencia colectiva. (p. 13)

La negociación a la que se refiere Alonso se da de manera particular en la práctica de live coding en red pues está determinada por el componente que la caracteriza: la interconexión. Es decir, la red de computadoras determina la forma de interacción puesto que permite un intercambio de código fuente en el que está plasmado el pensamiento del live coder.

La conexión entre el live coding en red y las prácticas de networking también se puede trazar a través de la improvisación libre, que en palabras de Chefa Alonso (2008) es una práctica de carácter colectivo en la que se establece una negociación social, y en la que

hay una exploración de la relación con otros músicos en un lenguaje abierto que no se sujeta a géneros musicales y que demanda una escucha atenta. Descripción que se relaciona con el concepto de networking en cuanto a que ambas exploran la relación entre individuos en prácticas colectivas. El live coding en red, en tanto caso de música en red e improvisación, encuentra resonancia en el concepto de networking al definirse como una práctica colectiva con base en un proceso que establece relaciones en el momento de la improvisación.

Otra idea que refuerza la propuesta de Alonso es lo que expresa Matthews (2012), quien dice que la improvisación grupal es una conversación entre los participantes que crea una "*plataforma social,* una cierta atmósfera de *entente* articulada por el esfuerzo de colaboración que requiere" (p. 24). Dicha conversación cobra forma en el sonido que resulta de la interacción en grupo y se establece por los distintos procesos sonoros en progreso que ocurren entre los individuos durante una improvisación colectiva.

Otra línea que puede trazarse en la conexión entre el live coding en red y las prácticas de networking es a través de la música en red y el movimiento hacker. Esta relación se puede plantear a partir del comentario de Brown y Bischoff (2002) cuando relatan que en los inicios de la música en red había una intención por explorar una tecnología de interconexión para la creación artística en grupo bajo el ideal del acceso libre y abierto a la información (véase §2.2). Esto no ocurre de manera aislada ya que, según Bazzichelli (2013), en los años setenta se desarrolla parte del movimiento hacker en California; movimiento que según Steven Levy (2010) —quien acuña el término ética hacker— estaba centrado en la creación de hardware para conectar personas, y al igual que los hacker predecesores de los años cincuenta y sesenta tenían "una filosofía de compartir, apertura, descentralización, y poner las manos en las máquinas a cualquier costo para mejorarlas, y mejorar el mundo"[91] (p. ix). La música en red, al desarrollarse en este momento y lugar —Área de la Bahía de San Francisco—, no escapa al espíritu de la época,

91 "It was a philosophy of sharing, openness, decentralization, and getting your hands on machines at any cost to improve the machines, and to improve the world".

por lo que se puede decir que la música en red comulga con los principios hacker.

Estas redes —red de computadoras y red de relaciones— no operan de manera separada, más bien en una relación que Sorensen (2007), con base en la teoría del actor red de Bruno Latour, llama *redes o ensamblajes socio-materiales* y están compuestas por conjuntos de entidades sociales o humanas y materiales o no-humanas; en el caso del live coding en red, la red de computadoras y la red de relaciones.

Para finalizar este capítulo, es posible concluir que los live coders de un ensamble construyen el sonido grupal a partir de la combinación de los eventos sonoros de la escritura individual que ejecuta la computadora en el momento de improvisar. En esta interacción se genera la red de relaciones entre los integrantes. Compartir el código abre la posibilidad de continuar el trabajo de todos, en cada nueva improvisación, a partir del código que funciona como contenedor de información y conocimiento, esto permite plantear que el live coding en red es una práctica en progreso.

Este capítulo ha expuesto la red de computadoras, la música en red y el networking. Así mismo, ha desarrollado un diálogo entre diferentes conceptos que trazan una línea que conecta al live coding en red con las prácticas de networking. Desde esta perspectiva, el siguiente capítulo da cuenta del live coding en red en el caso del grupo LiveCodeNet Ensamble.

3. Live coding en red en el caso de LiveCodeNet Ensamble

El capítulo anterior ha expuesto el concepto de red de computadoras y cómo, a partir de los años setenta, la interconexión se usa para crear música con computadoras de manera grupal. También presenta el concepto de networking y, a través de una serie de asociaciones con la improvisación libre, la música en red, los movimientos hacker y open source, se discute el concepto de live coding en red.

Con el fin de mostrar cómo ocurre el live coding en red, el presente capítulo expone el caso de LiveCodeNet Ensamble a partir del análisis de su práctica en el contexto del concierto y el ensayo, lo cual fue realizado mediante la observación de videos y audios de los conciertos realizados durante el periodo de la maestría, así como de entrevistas con algunos de sus integrantes. Así mismo, se realiza la descripción y análisis de la infraestructura que utiliza el ensamble para llevar a cabo su práctica.

Para estructurar el capítulo parto de la pregunta ¿cómo se configura la práctica musical de live coding en red en el caso de LiveCodeNet Ensamble? Esta pregunta arroja respuestas que describen el objeto de estudio en la práctica y dialogan con ideas y conceptos discutidos en los capítulos anteriores. Decidí ejemplificar el live coding en red con el caso mencionado pues es un grupo en el que participo como integrante, lo que me ha permitido realizar la investigación a través de interrogantes que surgen en la práctica y se problematizan con relación a los conceptos teóricos expuestos en los capítulos anteriores.

Esta sección de la tesis comienza por definir qué es LiveCodeNet Ensamble, para pasar al análisis y descripción de la infraestructura de hardware y software, luego a una discusión del acto de mostrar el código y cómo se estructura la forma de realizar live coding con el uso de la sincronía. El final del capítulo expone el ensayo y concierto en la práctica de LCNE.

3.1 LiveCodeNet Ensamble: una perspectiva grupal

LiveCodeNet Ensamble es un grupo de live coding mexicano cuyos integrantes crean música de manera conjunta mediante la escritura de código con sus computadoras conectadas a una red local. Originalmente integrado por José Carlos Hasbun, Hernani Villaseñor, Libertad Figueroa, Emilio Ocelotl, Eduardo Obieta y Katya Álvarez quienes provienen de disciplinas como la música, la comunicación, la sociología, la arquitectura y la ingeniería electrónica. El ensamble se formó en la Ciudad de México en octubre de 2013 en el contexto de las sesiones de live coding[92] y la serie de simposios internacionales de música y código[93] que el Taller de Audio[94] del Centro Multimedia organizó entre los años 2010 y 2014; eventos en los que también participé como co-organizador y live coder.

La motivación para formar LCNE fue explorar el live coding de manera grupal con el uso de una red de computadoras local para crear música. En una primera etapa, el trabajo del ensamble consistió en explorar el funcionamiento de la red de computadoras para compartir código, enviar mensajes de texto y sincronizar[95] las computadoras. Las primeras aproximaciones del ensamble se enfocaron en entender las posibilidades técnicas para tocar en grupo y compartir código con el uso del programa SuperCollider.

92 La sesiones de live coding fueron encuentros mensuales en formato de concierto en los que se presentaban entre 8 y 10 dúos de live coders. Los dúos se formaban en el momento y consistían en un participante que realizaba sonido y otro que realizaba imagen; ambos proyectaban su código a la audiencia junto a la música e imágenes que generaban. Cada par contaba con 9 minutos para realizar su presentación partiendo casi siempre desde cero en la escritura de código. La forma de participación era a través de una convocatoria abierta y una lista de correos electrónicos a la que respondían los interesados en participar. Véase <http://cmm.cenart.gob.mx/cartelera/2011/enero.html#livecoding>

93 El Simposio Internacional de Música y Código /*vivo*/ tuvo 3 ediciones durante los años 2012, 2013 y 2014 en los que se abordaron las temáticas de live coding, música en red y música y código. <http://vivo.cmm.cenart.gob.mx/>

94 El Taller de Audio (ahora Laboratorio de Audio) es un espacio en la Ciudad de México dentro del Centro Multimedia del Centro Nacional de las Artes donde se investigan tecnologías sonoras, análogas y digitales, que tiene una relación con distintas disciplinas artísticas.

Posteriormente, conforme el conocimiento técnico fue asimilado, la exploración del ensamble se centró en experimentar diversas estrategias para realizar live coding en red a partir de la improvisación.

Los integrantes que participaron de manera activa con el ensamble durante el periodo de la investigación —realizada entre los años 2014 y 2016— fuimos Emilio, Libertad, José Carlos y yo. Para definir al ensamble y describir cómo opera —además del análisis de la práctica y mi perspectiva desde la investigación como integrante— entrevisté en distintas ocasiones a los tres integrantes mencionados[96]. Las preguntas que me interesaba realizar eran cómo definían al ensamble, qué papel asumían en el momento de tocar en grupo, qué estrategias utilizaban para improvisar y cómo preparaban sus materiales. Sus respuestas arrojaron descripciones técnicas y estéticas de la práctica del ensamble, así como de diversas formas de interacción; confirmaron algunos supuestos de cómo es la práctica de live coding en red y dejaron ver que, en el caso particular del ensamble, algunas cosas no suceden como están planteadas desde la teoría del live coding. Por ejemplo, la práctica del live coding realizada por el ensamble no se enfoca en escribir el código fuente durante las presentaciones sino en reescribir código escrito de manera previa; y compartir el código entre los integrantes no necesariamente activa una dinámica de trabajo colaborativo enfocada en reescribir el código fuente de los demás.

Los integrantes coinciden en que definir a LCNE resulta difícil, que su organización y forma de hacer live coding es flexible y que estas características son atractivas para participar. Mencionan que la flexibilidad en la forma de tocar ha permitido

95 En este trabajo entiendo por sincronía cuando las computadoras conectadas a una red siguen un reloj común para realizar tareas de manera conjunta. Lee y Essl (2014) se refieren a la sincronía, en el live coding en red, como “tiempo compartido” [time sharing] (párr. 3). Los autores dicen que tocar juntos es tocar sincronizados y para lograr esto en un sistema de live coding se requiere “un reloj sincronizado entre las máquinas” [one synchronized clock between machines] (ibíd.).

96 Realicé dos entrevistas con cada integrante en diferentes momentos: al final de los ensayos, fuera de ellos y por correo electrónico. Procuré realizar las mismas preguntas en un tipo de entrevista semiestructurada.

al ensamble presentarse en lugares con distintos enfoques y públicos[97]. Al respecto, Emilio (E. Ocelotl, comunicación personal, 27 de agosto de 2015) comenta que pensar en los recintos donde se presenta o puede presentarse el ensamble le permite entender mejor cómo definirlo, por ejemplo, tocar en un festival o espacio especializado en música electrónica de baile o en uno de música electroacústica.

Por su parte, José Carlos (J.C. Hasbun, comunicación personal, 27 de agosto de 2015) lo define como un laboratorio donde se puede colaborar, aprender de los demás y compartir conocimientos. Este comentario se asocia a la idea expresada en el concepto de networking, con el que Bazzichelli (2008) propone que las redes de relaciones se crean en contextos de intercambio, compartición y experimentación artística. En este sentido, la red de relaciones que crean los integrantes del ensamble se gesta en las dinámicas alrededor de los ensayos y conciertos. Lo anterior puede entenderse desde lo que comenta Emilio (E. Ocelotl, comunicación personal, 27 de agosto de 2015) quien expresa que el ensamble puede pensarse como un conjunto de personas involucradas para resolver problemas relacionados con la composición de música: como insertar sonidos en una pieza; la operatividad: como planear y organizar ensayos y conciertos; y la técnica: como el desarrollo de una herramienta[98]. Esta forma de definir al ensamble, respecto a la construcción de redes de relaciones, amplía su entendimiento más allá del momento de la

97 El ensamble se ha presentado en espacios institucionales, universidades, festivales y bares de la Ciudad de México; y realizó una residencia virtual en el marco de la Muestra de Arte Sonoro e Interactivo In-Sonora 9 en Madrid, España. Según las respuestas de los entrevistados, estos sitios determinan la estética en el momento de tocar, con una orientación a la música electrónica de baile, a la electroacústica o una combinación de ambas. Se puede ver la historia de conciertos del ensamble en el siguiente vínculo: <https://livecodenetensamble.wordpress.com/fechas/>

98 Emilio se refiere al proyecto de desarrollo llamado Tweetensamble que comenzó por iniciativa de José Carlos, quien desarrolló en su totalidad un programa para escribir código en grupo y subirlo a Twitter. Algunos ensayos nos dedicamos a probar su funcionamiento y la meta era utilizarlo en una residencia virtual del microblog Twitter convocada para la Muestra de Arte Sonoro e Interactivo In-Sonora 9. La herramienta quedó en fase de desarrollo y la residencia la realizamos escribiendo código a través de correo electrónico.

improvisación.

Respecto a la estética los integrantes mencionan que LCNE improvisa y hace live coding, pero que resulta vago definirlo así. Hacen énfasis en la diversidad de estilos o géneros musicales como un componente característico, aunque mencionan que no se define bajo uno en específico. No obstante, nombran términos como techno, arte sonoro, experimentación, ambient, electroacústica, electrónica de baile, punk, live coding e improvisación.

Emilio comenta que en la estética del ensamble reside una parte importante de la música por computadora que comenzó a figurar en un entorno de música convencional que se acerca a discursos y técnicas de experimentación, respecto a lo anterior dice que "no se ha consolidado bien hacia qué lado vamos, si vamos del lado mucho más abstracto, más anclado en una tradición o si pretendemos tirarle a ese otro tipo de experimentación que cuaja bastante bien con un ámbito de la música de baile" (E. Ocelotl, comunicación personal, 27 de agosto de 2015). Emilio sugiere que incursionar en estéticas distintas genera una tensión que el ensamble mantiene sin resolver.

En este sentido, José Carlos comenta que el ensamble no tiene un estilo definido, dice que "tenemos herramientas como es el live coding, como es un lenguaje de programación, sin embargo, al decir live coding aún queda todo muy disperso, […] creo que bailamos de un estilo a otro, del techno al arte sonoro a la experimentación, un poquito de ambient, o sea, podemos hablar de géneros, sin embargo, lo interesante para mí es salirnos de esa retícula y que siempre pasamos por uno, vamos por otro" (J.C. Hasbun, comunicación personal, 27 de agosto de 2015). Al igual que Emilio, José Carlos sugiere un desplazamiento entre diversos géneros musicales que mantienen la estética del ensamble sin definirse. Este comentario es posible ligarlo a las características que Bailey (véase §1.4) otorga a la improvisación libre entre las que destaca la no adherencia a estilos o sonidos particulares, y que más bien atribuye a la identidad musical de cada persona que improvisa. Al respecto, José Carlos comenta que "de alguna manera el resultado sonoro de cada integrante representa algo de su personalidad" (J.C. Hasbun, comunicación por correo

electrónico, 23 de abril 2016).

Lo anterior se refuerza con lo que comenta Libertad quien dice que “no hay un estilo que lo defina y tal vez el estilo que lo define es la falta de estilo, […] estamos en ese lado que a veces resulta raro por lo mismo, por estar a veces tan indeterminado, […] me gusta esa indeterminación y no entrar en un género específico de música” (L. Figueroa, comunicación personal, 27 de agosto de 2015). Parece que esta aparente falta de estilo, marcada en la definición de Bailey cuando habla de la improvisación libre y expresada en la práctica que definen los integrantes de LCNE, acerca más la práctica que realiza el ensamble a la improvisación a partir de su estética que de una técnica específica.

3.2 La red de LCNE

La red de computadoras de LCNE está formada por computadoras, cables, routers y software; esta red presenta las siguientes características: es local, de difusión, con un modelo de comunicación par a par y una forma de conexión en estrella. Pero ¿qué quiere decir esto?, ¿qué implica en la práctica de LCNE?, ¿cómo se vincula con la red de relaciones que construyen sus integrantes?

En el caso de LCNE, la finalidad de usar una red de computadoras es realizar una práctica de creación musical en grupo en la cual se puede intercambiar y compartir el código que escribe cada integrante, enviar mensajes de texto para comunicar ideas y sincronizar los relojes de las computadoras, todo esto durante ensayos y conciertos.

Hay que recordar que Tanenbaum y Wetherall (2012), al hablar de una red de computadoras, se refieren a varias computadoras que se interconectan para realizar un trabajo, intercambiar información y compartir recursos. Esto se ve reflejado en la conformación de la red de LCNE, en la que cada integrante trabaja con su computadora y se comunica con los demás integrantes a través de la red tecnológica. Al respecto, José

Carlos define la red como “una red local simple que nos permite compartir datos, sincronizar el *clock* de los CPUs” (J.C. Hasbun, comunicación por correo electrónico, 23 de abril 2016) y agrega que esto permite “tener un performance interactivo entre los integrantes” (ibíd.). Esta configuración, a su vez, determina las dinámicas que ocurren entre los integrantes del ensamble. Por ejemplo, la sincronización de las computadoras permite estructurar el código en patrones rítmicos que suenan a tiempo.

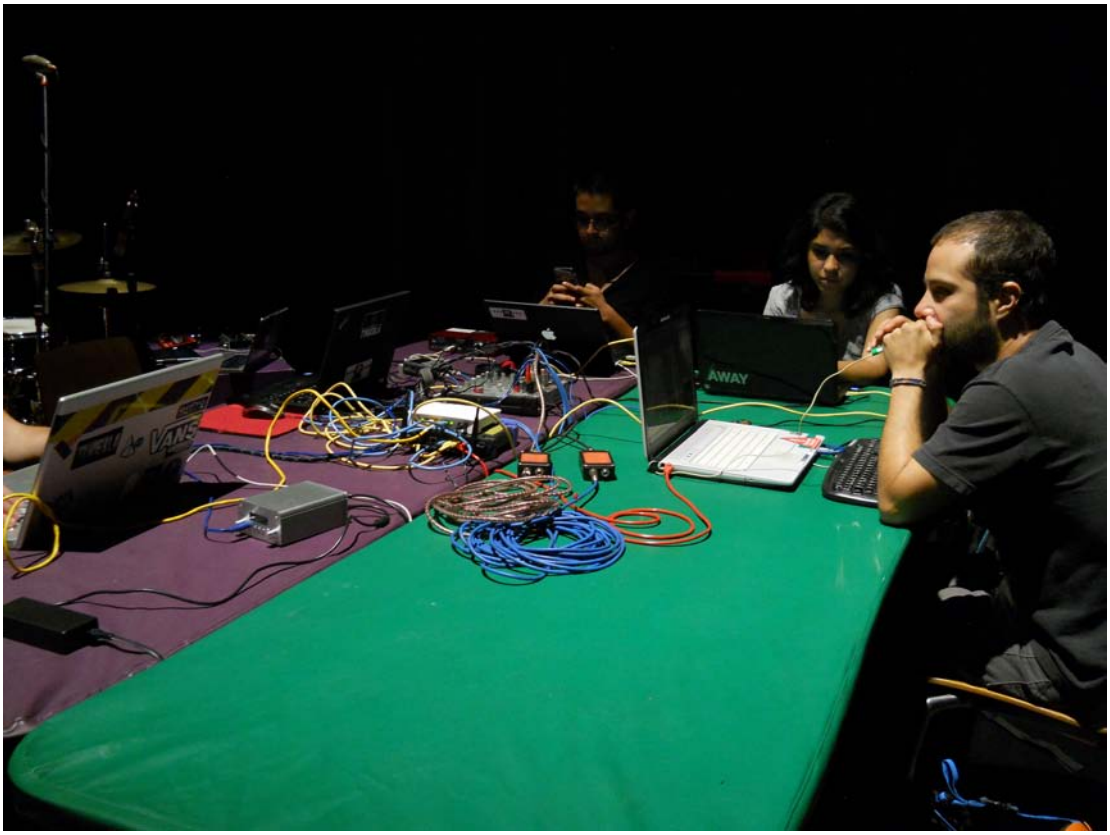


Figura 5. LiveCodeNet Ensamble y la red de computadoras. Fuente: Hernani Villaseñor (Archivo personal)

Por su parte, Libertad da una pista en la forma que los integrantes se relacionan a través de la red tecnología cuando comenta lo siguiente “soy muy consciente de la red cuando estamos tocando, […] de que hay algo que nos está ayudando a, tal vez, lograr un

resultado menos caótico” (L. Figueroa, comunicación personal, 23 de noviembre de 2015). Libertad se refiere a la interacción que el ensamble tiene a través de la tecnología que utiliza, en este caso la sincronía a la que se refiere José Carlos, cuya función es organizar en el tiempo los distintos eventos sonoros que generan los integrantes. Sin embargo, esto determina un tipo de estructura en la improvisación que depende de esta función.

3.2.1 Red local, de difusión, en estrella, par a par

La red de computadoras que utiliza LCNE es local, configuración que permite a los integrantes del ensamble interactuar en el mismo lugar durante cada ensayo o concierto. Una red de área local o LAN[99], según Barceló et al. (2004), surge de la necesidad de interconectar varias computadoras en una misma instalación para compartir recursos como impresoras o discos duros. Las redes locales, según estos autores, operan bajo la lógica de establecer una comunicación donde los paquetes de datos que salen de una computadora llegan al resto del equipo de manera simultánea mediante la “difusión con medio compartido”[100] (p. 30).

En cuanto a la distancia, escala y tamaño físico de una red de computadoras, Tanenbaum y Wetherall (2012) realizan una clasificación en: redes de área personal, redes de área local, redes de área metropolitana y redes de área amplia. A partir de esta clasificación es posible pensar una práctica de live coding en red conforme a la distancia y la escala del proyecto, es decir, las redes de computadoras en la música conectan a los integrantes de un grupo en dos situaciones respecto a la ubicación geográfica que ocupan: una es cuando se encuentran en el mismo espacio y la otra cuando están distribuidos en

99 LAN por sus siglas en inglés, *Local Area Network* o Red de Área Local.

100 Los autores explican el término en dos partes, *difusión* quiere decir que “los paquetes se difunden por todos lados” (Barceló et al., 2004, p. 30) y *medio compartido* que la difusión “se lleva a cabo sobre un medio común que las estaciones comparten” (ibíd.).

distintos puntos geográficos. En el primer caso, que es el que compete a este trabajo, una red de área local conecta las computadoras de los integrantes, mientras que en el segundo la red extiende su alcance a la distancia al utilizar Internet[101] para conectar las computadoras.

La forma que adquiere la red del ensamble, a partir de su configuración local, es una topología de estrella. Según Barceló et al. (2004) la topología es "la manera en la que se conectan las estaciones; […] la forma que adopta el medio compartido entre las mismas" (p. 55). La topología en estrella es una forma de conexión que, según los autores, caracteriza a la red local y "consiste en conectar cada ordenador a un punto central" (ibíd.). En el caso de LCNE el punto central es el aparato llamado router el cual realiza la conexión física de las computadoras de la red. El router, además de establecer una conexión física, tiene la función de encaminar y gestionar el tráfico de datos que circula en la red.

La figura 6 presenta una gráfica de la topología de estrella de la red de LCNE. Las flechas de la gráfica indican el flujo de datos entre las computadoras, muestran que cada una puede enviar y recibir información de todas las demás computadoras conectadas a la red. Estas flechas apuntan al router ya que los paquetes de datos primero pasan por este punto y desde ahí son dirigidos a cada computadora, lo cual se realiza con la ayuda de un programa, como puede ser SuperCollider, y diferentes protocolos[102]. El programa establece la comunicación entre las computadoras mientras que los protocolos distribuyen los datos entre las computadoras conectadas en la red y permiten que éstas se entiendan, pues la red está conformada por diferentes sistemas operativos y modelos de computadoras.

101 Por ejemplo, el entrono de código creativo Gibber <http://gibber.cc/> funciona desde un navegador de Internet. Según su manual, puede ser utilizado en sesiones de edición colaborativa de código y en presentaciones en red. Otro programa que puede ser utilizado para una práctica a la distancia es el sistema de live coding en red Extramuros <https://github.com/d0kt0r0/extramuros>, el cual, según su sitio web, es un sistema diseñado para escribir código en espacios de memoria compartidos. Este sistema permite escribir código con diferentes lenguajes de live coding, por ejemplo, SuperCollider o TidalCycles.

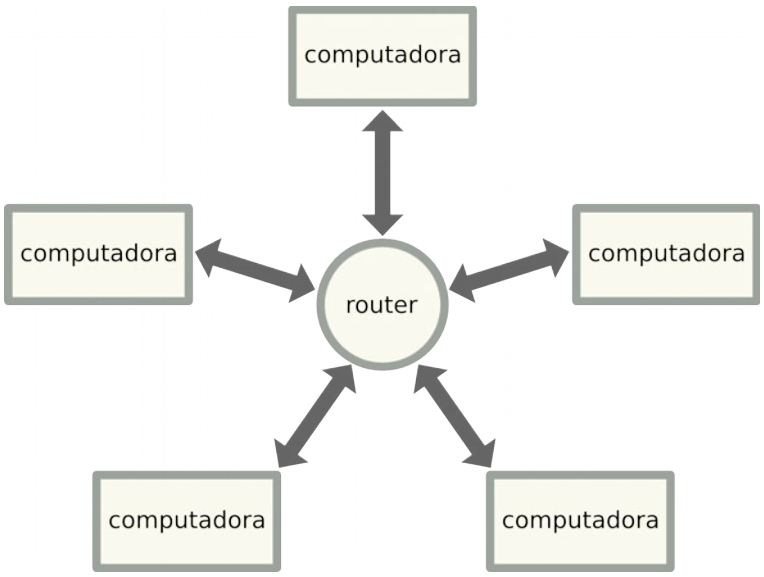


Figura 6. Topología de estrella

Para Alexander Galloway (2004) un protocolo es “el principio de organización nativa de las computadoras en redes distribuidas” (p. 3), el autor entiende por red distribuida una estructura sin centro. Galloway dice que éste es el lenguaje común de las computadoras conectadas en red, que “sin un protocolo compartido, no hay red” (p. 12). Así

102 En este caso los protocolos TCP/IP y OSC. TCP/IP es un par de protocolos que trabajan juntos para la comunicación en Internet. El primero, es el protocolo de control de transmisión (*transmission control protocol*) y el segundo es el protocolo de Internet (*Internet protocol*). Según O'Regan (2013) ambos fueron desarrollados por Vint Cerf y Robert Kahn dentro de DARPA (*Defense Advanced Research Projects Agency*). El autor cuenta que el protocolo TCP, desarrollado en 1974, es un estándar de comunicación entre computadoras que permite a varias redes interconectarse para funcionar como una sola red, pues está diseñado para la transmisión de datos entre diferentes computadoras, routers y redes. Más tarde se desarrolló el protocolo de TCP/IP. En el mencionado estándar, el TCP rompe los datos en paquetes para luego reconstruirlos, mientras que el IP se encarga de enviarlos a través de la red dándoles ruta y dirección; de esta manera se envían correos electrónicos o archivos digitales. Por otro lado, OSC (*open sound control*) es un protocolo que, según su sitio web, está diseñado para comunicar computadoras, sintetizadores y aparatos multimedia optimizados para la tecnología de interconexión actual.

mismo, menciona que el protocolo son reglas y recomendaciones que describen un estándar técnico.

La red, además, está configurada bajo un modelo de comunicación *par a par* o *igual a igual* que permite una conexión de tipo todos con todos. Tanenbaum y Wetherall (2012) lo describen como un modelo en el que cualquier individuo conectado a la red se puede comunicar con uno o más individuos puesto que “no hay división fija en clientes y servidores” (p. 6). En el modelo cliente-servidor todas las computadoras de la red, llamadas clientes, dependen de una computadora central o servidor que contiene toda la información. Contrario al modelo cliente-servidor, en el modelo par a par cada computadora es independiente y contiene su propia información; según los autores, no hay una base de datos central. Bajo este modelo, cada integrante interactúa desde su computadora, a través de la red, con los demás integrantes conectados. Tanenbaum y Wetherall comentan que “la comunicación de igual a igual se utiliza con frecuencia para compartir música y videos” (ibíd.). De igual forma, una red par a par puede utilizarse para compartir código fuente.

Así mismo, la red de LCNE está conformada para trabajar como una *red de difusión*, en la que, según Tanenbaum y Wetherall (2012), “los paquetes que envía una máquina son recibidos por todas las demás” (p. 15). Esto permite al código fuente llegar de manera simultánea e indistinta a todas la computadoras interconectadas en el momento que es compilado o interpretado debido a que el software que gestiona la red canaliza los datos a la *dirección de difusión* de cada computadora conectada. La dirección de difusión es una dirección IP especial marcada con el número 255.255.255.255, que según Barceló et al. (2004) es una dirección de destino utilizada para transmitir paquetes a todas las computadoras que se localizan en una misma red local.

A continuación presento un fragmento del software[103] utilizado por LCNE para conectarse en red. En este fragmento se observa la configuración de la red de difusión que posibilita a los integrantes del ensamble enviar código fuente a todas las computadoras conectadas a la red.

103 El código del software de conexión se muestra entero en la sección Anexos.

```
(
q = ();
NetAddr.broadcastFlag = true;
q.addrs = (0..7).collect { |x|
            NetAddr("255.255.255.255", 57120 + x)
};
q.sendAll = { |q ... args|
            q.addrs.do { |addr|
                        addr.sendMsg(*args)
            }; ""
}
);
```

Figura 7. Fragmento del código de conexión

El código de la figura 7 muestra dos cosas: la primera está en la tercer línea de código en la que aparece el objeto `NetAddr` con el mensaje `broadcastFlag` como verdadero indicando que el sistema funciona como una red de difusión[104]; y la segunda está en la línea 5 donde se ve la dirección IP de difusión `"255.255.255.255"` a la que es enviada la información. Esto quiere decir que los paquetes de información enviados dentro de la red serán recibidos por todas las computadoras del ensamble que estén conectadas. La forma de conectar las computadoras a la red se realiza en dos pasos: primero se conecta cada computadora al router con un cable ethernet[105] y luego se debe compilar el código de conexión en cada computadora.

104 *Broadcast* es el término en inglés para Red de difusión.

105 Ethernet es un protocolo de red que, según Barceló et al. (2004), es el más utilizado en las redes de área local.

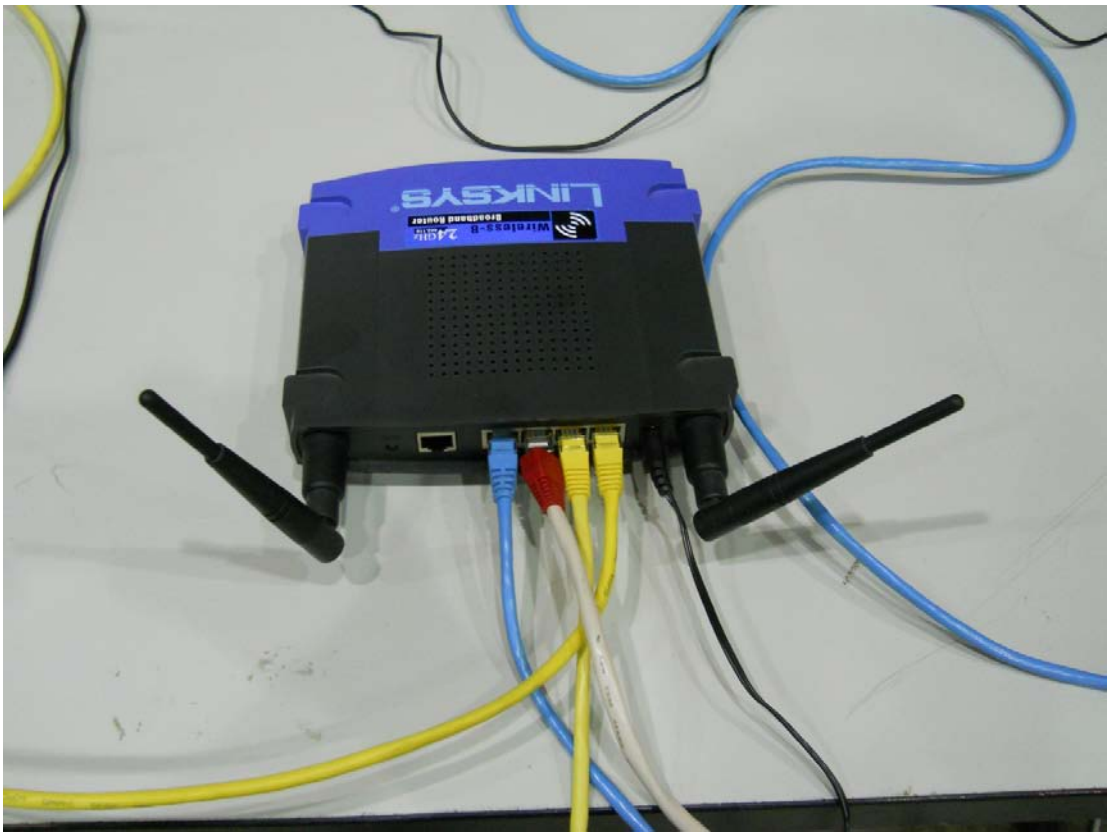


Figura 8. Conexión de cables ethernet al router. Fuente: Hernani Villaseñor (Archivo Personal)

Hasta este punto se ha discutido la infraestructura que permite la circulación de datos en una red, sin embargo, esta descripción queda incompleta sin el análisis de la configuración de las señales de audio de las computadoras de cada integrante. A continuación se explica cómo se gestiona el sonido en el momento que LCNE realiza live coding y se establece una topología de esta conexión para compararla con la conexión en red de las computadoras.

3.2.2 El sonido en la red de LCNE

Al realizar live coding, durante un concierto o ensayo, los integrantes de LCNE amplifican

el sonido de sus computadoras. La manera de hacerlo es conectando las computadoras a una mezcladora que envía la señal a un par de bocinas configuradas en estéreo. Este punto es crítico pues el volumen de sonido que cada computadora emite se concentra en el aparato mezcladora, un punto de paso que atenúa o aumenta las diferentes señales de audio con base en el criterio de quien la opera. Las señales de audio de las computadoras se acomodan en la mezcladora durante la prueba de sonido; una vez niveladas el concierto se desarrolla con los volúmenes intactos o realizando pequeños ajustes. Durante los conciertos, cada integrante controla su volumen desde su computadora y de esta manera busca un lugar en el sonido generado grupalmente a partir de escuchar lo que va resultando. La estrategia de acomodar los volúmenes de una forma, aparentemente neutral, se ve afectada por diversas circunstancias como la tarjeta de sonido de cada computadora y el tipo de sonidos que genera cada integrante. En este sentido, escuchar es lo que permite a cada integrante del ensamble guiarse dentro del sonido grupal emergente. La escucha se divide en entender lo que cada integrante emite y lo que resulta de la combinación de todos.

Acerca de la escucha en la actividad el ensamble Libertad comenta “[escuchar] me da más idea de mi contexto, me da más idea de lo que estoy haciendo, pienso que es elemental escuchar qué están haciendo los demás” (L. Figueroa, comunicación personal, 23 de noviembre de 2015). El comentario de Libertad resalta la escucha, y es que gran parte de la argumentación sobre la realización de live coding se basa en la visibilidad y legibilidad del código, pero ¿cómo se escucha lo que produce el código? y ¿cómo se escucha de manera grupal? La práctica de live coding implica ver código fuente y leerlo pero también escuchar lo que produce para activar la conversación que se crea en el momento de la improvisación grupal. Asimismo, Libertad describe cómo construye su sonido a partir de la escucha “no pienso mucho a la hora de tocar, […] escucho más de lo que pienso, […] los escucho y entonces intento ubicarme en algún punto sonoramente a partir de lo que yo ya había trabajado antes […]. A partir de lo que yo ya sé que traigo, que ya sé cómo suena, intento ubicarlo” (ibíd).

Un problema al que se enfrentan los ensambles que amplifican las señales de sus

computadoras en un sistema de sonido compartido es la concentración de éstas en un solo aparato, en contraste con la distribución de la información que ocurre en la configuración de las redes par a par. Es decir, el concepto de apertura que genera la práctica en una red de computadoras de este tipo entra en tensión en el momento de amplificar el sonido. En este sentido, algunos grupos optan por incorporar un sistema de audio independiente para cada computadora, de tal forma que cada integrante es responsable de generar su propio sonido[106], aunque esto depende del acceso a infraestructura y del lugar donde se realiza el concierto.

Para comparar la red de computadoras y la conexión de sonido de LCNE se puede caracterizar la última como una topología distinta a la forma de estrella que presenta la interconexión de computadoras del ensamble. Con este fin me remito a la "topología de flor" propuesta por Gil Weinberg (2005, p. 34) quien describe el diseño de redes de computadoras para la música y sus diferentes topologías[107]. La topología de flor descrita por Weinberg es una conexión donde las computadoras se conectan a un punto central, y que el autor describe como una red centralizada y sincrónica. En el caso de este trabajo, tomo la conexión central descrita en esta red para ilustrar la conexión a la mezcladora de audio. A diferencia de la topología de estrella, la cual distribuye o encamina datos e información desde un punto central hacia todos los puntos conectados a la red, la topología de flor concentra las señales en un punto —como es el caso de la mezcladora.

En la siguiente gráfica se observa la conexión de las computadoras a la mezcladora de sonido representada por la topología de flor. En este caso las flechas representan las

106 El grupo Power_Books UnPlugged (Rohrhuber et al., 2007) resalta la importancia de co-existir en el espacio acústico como grupo, para esto PB_UP utiliza las bocinas de cada computadoras portátil como única amplificación. Otro ejemplo es la orquesta de laptops PLOrk, como se puede leer en su sitio web cada integrante toca con su computadora y una bocina hemisférica para emular las proyección de sonido de una orquesta tradicional.

107 Weinberg (2005) menciona las topologías de flor [flower], de carretilla [wheelbarrow], de estrella [star] y de escaleras [stairs]. Para compararlas véase el artículo *Interconnected Musical Networks: Toward a Theoretical Framework*.

señales de audio de las computadoras viajando en una sola dirección, las cuales llegan a un punto donde se concentran y la interacción se desplaza al sonido resultante que emerge de las bocinas.

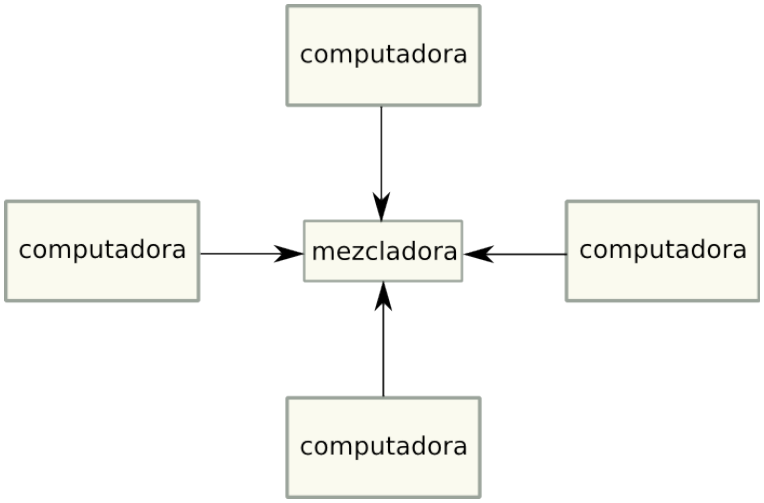


Figura 9. Topología de flor aplicada a la conexión de audio

Topología de estrella	Topología de flor
Forma de conexión de la red de computadoras	Forma de conexión de las señales de audio de las computadoras
Distribuye los datos	Centraliza las señales
Los datos son gestionados por el aparato router	Las señales de audio son gestionadas por el aparato mezcladora

Tabla 3. Comparación de la topología de estrella aplicada a la red de computadoras y la topología de flor a la conexión de audio en el caso de LCNE

Lo que quiero mostrar con este ejemplo es que la negociación del sonido en un grupo de live coding no solo depende la de conexión en red de las computadoras, también radica en la forma de organizar las señales de sonido, en escuchar a los demás y no solo en la visualización del código. Sin embargo, a partir de compartir e intercambiar código se

genera un discurso que a veces no se traslada a la generación de sonido en grupo. En este sentido, hay un espacio de negociación que se encuentra en el sonido que escuchamos grupalmente.

Shelly Knotts (2015) menciona que la música en red tiene un potencial político que se revela a partir de los sistemas que escogen los ensambles que practican música en red. Lo que comenta la autora es en relación al "potencial de democracia"[108] (p. 49) que presentan distintos tipos de ensambles musicales; en el caso de la música en red se refleja por el tipo de "sistema" (p. 47) con el que decide trabajar cada ensamble[109]. Estas ideas son importantes en la música en red ya que ponen de manifiesto que la red de computadoras —o sistema como lo llama Knotts— se configura con base en las necesidades y pensamiento de un grupo, lo que determina la forma en la que hacen música.

Algo que es importante remarcar es que la configuración de sonido del ensamble, si bien contrasta con la conexión en red, define una manera de interactuar de los integrantes mediada por el sonido. La interacción a partir de la escucha sucede en la combinación del sonido de todos los integrantes. El sonido resultante es el punto donde la escucha se vuelve colectiva.

3.2.3 SuperCollider: el software de la red y práctica de LCNE

Para llevar a cabo la interconexión y realizar live coding, el ensamble utiliza el programa SuperCollider, el cual permite escribir código para generar sonido en el contexto de live coding, conectar las computadoras en red y sincronizarlas. Al utilizar el mismo software y lenguaje de programación los integrantes escriben y leen código que les permite entender qué hacen los demás, lo que facilita compartir e intercambiar información y conocimientos

108 "The democratic potential".

109 Para ver el potencial de democracia propuesto por Knotts (2015) para distintos grupos, véase la Tabla 1 del artículo *Changing Music's Constitution: Network Music and Radical Democratization*.

a partir del código. Por otro lado, al utilizar el mismo software es posible establecer la conexión y comunicación entre las computadoras de la red. Pero, además de la legibilidad y conectividad del código ¿por qué utilizar el programa SuperCollider?

El uso de SuperCollider en la práctica de LCNE obedece a un factor educativo ya que es el programa que se enseñó entre los años 2007 y 2014 en el Centro Multimedia, lugar en el que se formó el ensamble y donde los integrantes han estudiado, trabajado o realizado servicio social[110]. Además, el programa tiene varios atributos, aparte de los antes mencionados, que lo hacen viable para realizar esta práctica: es software libre[111], se ha desarrollado bajo los principios del código abierto, es un lenguaje de programación de alto nivel, es interpretado y es multiplataforma.

El software que conecta la red del ensamble está formado por dos piezas, una es un código[112] de SuperCollider que fue escrito por Alberto de Campo y Julian Rohrhuber en el taller *Algoritmofagia y Pensamiento público - parte 2*[113] durante el Simposio Internacional de Música y Código: Vivo 2012. El código fue retomado y modificado por el ensamble para adaptarlo a su práctica. La otra pieza de software es una extensión de SuperCollider

110 El Taller de Audio —Laboratorio de Audio desde 2015— del Centro Multimedia (CMM) impartió una serie de cursos anuales de SuperCollider entre los años 2007 y 2014. El primer registro que aparece en el sitio web del CMM es *SuperCollider 3 para principiantes* en noviembre del 2007 y fue impartido por los integrantes del Taller de Audio Ernesto Romero y Ezequiel Netri. En estos cursos participé como alumno y a partir de mi ingreso al Taller de Audio, en el año 2010, también impartí, junto a Ernesto Romero, el curso de SuperCollider. Más tarde, el curso se convirtió en *Música por Computadora*, curso modular que cubría los temas de composición algorítmica, live coding, música y programación. *Música por computadora* lo impartimos por última vez Alberto Cerro y yo en mayo del 2014 con lo que finalizó un periodo de 7 años de enseñanza del programa en el CMM y en el que se formó una amplia comunidad de usuarios de los cuales varios participan, desde entonces, en actividades relacionadas con el live coding en la Ciudad de México.

111 SuperCollider fue publicado por su creador como software libre en el año 2002 bajo una licencia GNU General Public License. Véase el sitio de SuperColldier en Internet: <http://supercollider.github.io/>.

112 El software de conexión puede verse en el anexo de este trabajo y descargarse del repositorio GitHub en la siguiente dirección: <https://github.com/hvillase/lcne/blob/master/01start-code.scd>.

llamada MandelHub[114] que es empleada por el ensamble para sincronizar los relojes de las computadoras.

Para escribir código que la computadora pueda interpretar y ejecutar en el momento, el ensamble utiliza la extensión de SuperCollider *Just-In-Time Programming Library* o JITLib[115], la cual, según Rohrhuber y De Campo (en Collins, 2011), "provee un entorno experimental y de improvisación" (p. 207). El uso de esta extensión y del programa SuperCollider por parte los integrantes del ensamble permite realizar una práctica conjunta en la que todos entienden el mismo lenguaje de programación aunque, por otro lado, la actividad resulta en cierta medida estandarizada.

Al respecto, Tanenbaum y Wetherall (2012) comentan que "para permitir que varias computadoras se comuniquen entre sí se requiere una gran cantidad de estandarización, tanto en hardware como en software" (p. 73). Esto quiere decir que las computadoras conectadas a una red necesitan entender los mismos protocolos para que las computadoras puedan vincularse y comunicarse en una red. SuperCollider, como el software encargado de la conexión de la red de LCNE, entiende las reglas de distintos protocolos como TCP/IP y OSC (véase §3.2.1), que permiten conectar las computadoras y gestionar el envío y recepción de datos entre ellas con la ayuda del router.

La práctica de live coding en red requiere que sus practicantes tengan

113 La información con la que se anuncia este taller dice: "No es fácil trabajar juntos, compartir herramientas y calendarios. Es aún más difícil comenzar a pensar juntos, compartir conceptos, darle la vuelta a los pensamientos, jugar con las ideas. Hacer *live coding* en red con procesos de sonido es una forma certera de perder el control de lo que acabamos de pensar hace un minuto; pero entonces: ¿por qué uno querría mantener el control sobre una conversación polifónica de ideas?". Para revisar la información completa del taller véase <http://vivo.cmm.cenart.gob.mx/vivo2012/talleres.html>

114 MandelHub es definida en la plataforma de desarrollo GitHub como una extensión de SuperCollider para realizar presentaciones colaborativas y sincronizadas con *laptops*. Forma parte de la librería BenoitLib, programada por Patrick Borgeat para el grupo Benoit and the Mandelbrots. BenoitLib se puede descargar del siguiente sitio: <https://github.com/cappelnord/BenoitLib>

115 Según Collins et al. (2003) la extensión JITLib de SuperCollider fue desarrollada por Julian Rohrhuber para facilitar el live coding durante una presentación.

conocimientos especializados, y sus equipos presenten un grado de estandarización, ambas cosas —especialización y estandarización— se logran a partir del lenguaje de programación; las computadoras entienden una serie de protocolos, mientras que los integrantes entienden el código que todos escriben. Sin embargo, aunque el software utilizado es el mismo, esto no quiere decir que los conocimientos de los integrantes son estandarizados, pues la forma en que escribe y se aproxima cada integrante al código es distinta. Esto, más bien crea una diversidad en las formas de hacer live coding con el mismo programa.

Aunque este trabajo expone a SuperCollider como un software que posibilita realizar live coding, es pertinente mencionar que existen varios lenguajes con los que se puede realizar esta práctica[116]. Entre los más utilizados se encuentran Ixi Lang, Extempore, Overtone, LiveCodeLab, Gibber, TidalCycles, Sonic Pi, FoxDot, Pure Data y ChucK de los cuales me voy a referir a TydalCycles y Sonic Pi por su repercusión actual en el live coding, ChucK por el concepto de programación *on-the-fly* que surgió a la par del término live coding y Pure Data por su método de programación distinto a la escritura de código textual.

TidalCycles[117] es un lenguaje para realizar live coding a partir de patrones rítmicos, en sí no produce sonido; funciona como secuenciador de muestras de audio y se interconecta con SuperCollider para trabajar con sonidos sintetizados. La particularidad de este lenguaje, como menciona su sitio web, es que permite crear patrones complejos de manera rápida y sencilla. TidalCycles es desarrollado desde 2009[118] por Alex McLean con la ayuda de diferentes colaboradores, además cuenta con una creciente comunidad de usuarios.

Sonic Pi[119], como menciona su sitio web, es un instrumento musical de código

116 Para consultar una lista amplia de programas con los cuales es posible realizar live coding véase la entrada en Wikipedia <https://en.wikipedia.org/wiki/Live_coding#Notable_live_coding_environments>

117 <https://tidalcycles.org/>

118 <https://en.wikipedia.org/wiki/TidalCycles>

119 <http://sonic-pi.net/>

abierto para realizar live coding. Es desarrollado por Sam Aaron y un equipo de colaboradores en el Laboratorio de Computo de la Universidad de Cambridge. Además de instrumento musical, Sonic Pi es un proyecto de educación para aprender programación de código a través de la música.

Chuck[120] es un lenguaje desarrollado por Ge Wang y Perry Cook desde el 2002 bajo el concepto *on-the-fly programming*. Este programa fue utilizado en los inicios de la orquesta de laptops de Princeton PLOrk[121]. Zmölnig y Eckel (2007), mencionan que ChucK quizá fue el primer lenguaje musical de computadora diseñado para hacer live coding.

Por último, Pure Data[122] presenta un método de programación visual distinto al de los lenguajes textuales. Aunque Pure Data no está diseñado para el live coding, su método de programación es mencionado por Collins et al. (2003) como una forma de realizar esta práctica; los autores hacen referencia a la "construcción de parches en vivo"[123] (p. 321). Los autores mencionan que "los usuarios de Reaktor, PD, o MAX/MSP han editado la estructura de la señal gráfica mientras que sus parches corren"[124] (ibíd). Zmölnig y Eckel (2007) mencionan que Pure Data contiene extensiones para realizar live coding de sonido e imagen, también refieren que los lenguajes gráficos de música por computadora ofrecen una representación visual del código fuente que los hacen más accesibles y familiares a la lectura de la audiencia a diferencia de los lenguajes de código textual.

120 <http://chuck.cs.princeton.edu/>

121 <http://plork.princeton.edu/people.html> Como se puede observar en este sitio web, en sus inicios la orquesta de laptops PLOrk fue dirigida por Perry Cook, desarrollador junto con Ge Wang, del programa ChucK.

122 <https://puredata.info/>

123 "LIVE PATCH BUILDING". La palabra *patch* es utilizada en la programación visual, como el caso de Pure Data o Max, para referirse al documento que contiene la programación. A diferencia de un programa textual que presenta código, un *patch* muestra objetos conectados por líneas que representan cajas conectadas por cables.

124 "Reaktor, PD or MAX/MSP users have edited the structure of the signal graph as their patches runs".

3.2.4 Sincronía, patrones e interacción con la red

Una forma en la que los integrantes de LCNE se organizan para tocar juntos es sincronizando el tiempo de los relojes de sus computadoras, situación común entre los grupos que realizan live coding en red con base en patrones rítmicos. Lee y Essl (2014) especifican que el tiempo es un dato[125] que se comparte en los sistemas de redes de live coding colaborativo. Como se menciona en el apartado anterior, para sincronizar sus computadoras, LCNE utiliza la extensión de SuperCollider llamada MandelHub[126]. Cuando las computadoras están sincronizadas, las acciones que ejecutan suceden de manera coordinada a lo largo del tiempo de la improvisación. La sincronización de las computadoras permite a los integrantes del ensamble colocar cada evento sonoro bajo un reloj común que establece el momento en el que cada acción descrita por el código fuente se ejecutará. Este reloj administra el tiempo en el que se generan los distintos sonidos, lo que posibilita incorporar cada evento sonoro al sonido grupal emergente acomodándolo en una rejilla temporal.

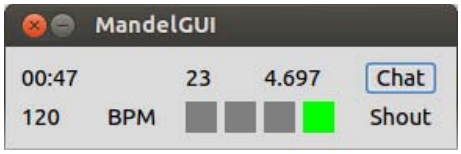


Figura 10. Ventana de MandelHub que indica el tiempo de sincronía. Toma de Pantalla

125 Los autores definen cinco datos compartidos en una red: *time sharing* o sincronización; *code sharing* o texto/representación del programa; *program-state-sharing* o estados, variables, objetos, memoria; *acces-control; communication-facilitation* o chat.

126 Como se puede observar en el código fuente de la extensión MandelHub (https://github.com/cappelnord/BenoitLib/blob/master/MandelHub.sc) ésta se rige por un reloj que marca el tiempo en pulsos [beats], acepta cambiar el tempo, establece una resolución [quant] de cada tiempo en 16 fracciones y permite ver de manera gráfica un compás de cuatro tiempos. Con estas características es posible crear música sincronizada bajo diferentes métricas y tempos.

La sincronía permite a los integrantes del ensamble organizar los eventos sonoros de manera rítmica. Una forma de hacerlo es escribir el código en patrones (véase figura 4), los cuales son interpretados por la computadora como *loops* de sonido y acomodados en el tiempo por el reloj de la sincronía. De esta manera —en sincronía y organizando el código en patrones— los integrantes del ensamble colocan, ordenan y estructuran los sonidos que van describiendo en el código para construir el sonido grupal emergente.

Cada integrante decide en qué momento compilar o interpretar un fragmento de código, sin embargo, es la sincronía la encargada de acomodar el sonido con base en el reloj común al que están vinculadas las computadoras. En este momento, la decisión de colocar cada evento sonoro se distribuye entre los integrantes y la red de computadoras. La sensación de sincronía nos deja entender que la red de computadoras está activa, que *algo* externo a la interacción humana mantiene el tempo del grupo y le ayuda a tocar en conjunto. Ese algo es una función programada, una comunicación entre las computadoras con la que interactuamos en el momento de improvisar y a la cual delegamos la tarea de mantenernos coordinados temporalmente en nuestra creación musical. En este sentido, Andrew Brown plantea que “las máquinas pueden ser co-creativas con humanos que trabajan en entornos culturales”[127] (2016, p. 180), mientras que Martin y Gardner (2015) sugieren que “la computadora puede asumir la responsabilidad de llevar el tiempo”[128] (p. 36).

A partir de lo anterior, es posible argumentar que la sincronía, como vínculo entre los integrantes del ensamble y la red de computadoras, participa de manera activa junto al grupo de live coders en la construcción del sonido de la improvisación. Desde esta perspectiva, la sincronía tiene un papel en la definición de la estética de la música creada y en la organización de la estructura musical.

127 “machines can be co-creative with humans working within cultural settings”.

128 “the computer can assume the responsability of keeping time”.

3.2.5 Compartir el código en la acción de dejar ver

McLean (2011) menciona que "la presentación arquetípica de live coding involucra programadores que escriben código en un escenario, con sus pantallas proyectadas para la audiencia y su código dinámicamente interpretado para generar música o video"[129] (p. 129). Al igual que otros autores (Collins et al., 2003; Wang y Cook, 2004; Ward et al., 2004), McLean resalta uno de los aspectos que aportan las presentaciones de live coding a los conciertos de música por computadora: mostrar cómo se va escribiendo el código fuente durante la presentación. Exponer a la audiencia la actividad de escribir código se hizo explícita en el año 2004 cuando el colectivo TOPLAP[130] pronunció en un punto de su manifiesto[131] "El oscurantismo es peligroso. Muéstranos tu pantalla"[132] (Ward et al., 2004, p. 247), con lo que demandaban transparentar la práctica de live coding en el momento de una presentación.

Lo anterior, según Ward et al. (2004), surge como respuesta a la controversia[133] que había en torno a la música por computadora tocada en vivo a principios de los años dos mil,

129 "The archetypal live coding performance involves programmers writing code on stage, with their screens projected for an audience, their code dynamically interpreted to generate music or video".

130 TOPLAP es, como lo indica el sitio web <http://toplap.org/about>, una organización dedicada a explorar y promover el live coding desde el año 2004. Se puede leer en el sitio que sus intereses giran en torno a la improvisación de música electrónica y video dentro del contexto del live coding. Plantean que el live coding se desarrolla bajo una diversidad de estilos y se practica en diversos lugares, desde salas de concierto hasta clubs de música electrónica. TOPLAP apela por la inclusión y el acceso para todos, por lo que promueve el software libre y alienta al aprendizaje de diferentes softwares para realizar live coding consultando a las comunidades de sus usuarios.

131 Para una lectura completa del manifiesto véase <http://toplap.org/wiki/ManifestoDraft>

132 "Obscurantism is dangerous. Show us your screens".

133 Ward et al. (2004) hacen referencia a textos escritos por Roger Dean y Andrew Schloss, en los que critican el hecho de proyectar las pantallas durante los conciertos, la poca gestualidad y causalidad de la música por computadora, así como la falta de formación musical de programadores informáticos que realizan música por computadora para tocar en vivo.

en la que se criticaba la falta de gestualidad y causalidad de ésta, y el aparente oportunismo de los programadores para tocar música en vivo sin tener una formación musical. De esta manera, los practicantes de live coding comienzan a poner mayor énfasis en proyectar el proceso de escritura del código para contrarrestar lo que reconocen como una "restricción física de la interfaz del teclado de computadora"[134] (Ward et al., 2004, p. 247). Pero ¿qué implica mostrar el código?, ¿qué es lo que muestra?

La proyección del código muestra a la audiencia el momento en que aparece cada letra y símbolo que teclea el live coder. Esta acción expone el pensamiento del live coder, cómo construye sus ideas en la escritura. En este punto, hay que mencionar que la apertura del código no viene de la práctica de live coding; mostrar el código tiene su origen en la cultura del código abierto y software libre que permiten el acceso al código fuente de un programa con el objetivo de que otros usuarios puedan estudiarlo, modificarlo y mejorarlo. En este sentido, Julien Ottavi (2008) plantea que el despliegue del software libre que ha ocurrido sobre prácticas artísticas ha permitido descubrir, en el arte, conceptos como:

> La lógica del código abierto, de compartir y distribución abierta, la libertad de modificar fuentes, nuevas formas de compartir, nuevas formas de aprender o de transformar el trabajo de otros, trabajando en red o colectivo, cuestionando derechos de autor y otras formas propietarias[135]. (Ottavi, 2008, p. 29)

En el live coding, el código se abre para mostrarse en dos situaciones: la primera, como se menciona, al exponer el proceso de escritura a una audiencia; la segunda al compartir el código con los integrantes de un ensamble que están conectados en red. En

134 "physical restriction of the computer keyboard interface".

135 "the logic of source code, of sharing and open distribution, the freedom to modify sources, new modes of sharing, new ways to learn or to transform other's people work, working in a net or a collective, questioning copyrights and other property rights".

ambos casos, dejar ver el código fuente a los demás muestra lo que sabemos hacer en la práctica de live coding. Ottavi (2008) habla de una continuidad, de un trabajo en proceso, y en progreso; el autor sugiere que compartimos lo que somos y lo que pensamos, que se borra la gran figura en favor del núcleo, en un "renacimiento del autor colectivo"[136] (p. 31). Lo mencionado por Ottavi tiene resonancia en el concepto de networking de Bazzichelli (véase §2.3), pues ambos plantean prácticas que se desarrollan de manera progresiva, en procesos colectivos.

Sin embargo, compartir el código requiere, en parte, de un entendimiento del lenguaje de programación, no tanto de la audiencia pero al menos de nuestros pares. En este aspecto, Ward et al. (2004) comentan que no es necesario que la audiencia entienda el código para apreciarlo, así como no lo es tocar una guitarra para disfrutar un concierto de este instrumento. Sin embargo, el código no es del todo abstracto, como se ha mencionado (véase §1.2), el código contiene palabras reconocibles, en inglés, que le dan pistas al espectador. Es distinto en una situación de un concierto de live coding en red en la que se muestra el código a pares que entienden el lenguaje con la intención de intercambiar y compartir. Pero ¿cómo sucede el intercambio y la compartición?, ¿cómo acceden los integrantes de LCNE al código que se va produciendo durante la improvisación?

El código que circula en la red aparece en las pantallas de las computadoras en una ventana del programa SuperCollider llamada History. En esta ventana el código llega en una secuencia temporal parecida a un chat, es decir, las líneas de texto del código aparecen con el nombre de quien las envió y el tiempo en que lo hizo. Cuando un integrante declara un fragmento de código —presiona una combinación de teclas para indicarle a la computadora que compile o interprete el código seleccionado— aparece de manera inmediata en la pantalla de cada computadora conectada a la red. Lo anterior es debido al diseño de difusión antes mencionado (véase §3.2.1) que permite circular la información sin restricciones y de forma simultánea en todas las computadoras interconectadas. En la figura 11 se observa la ventana History, en cuya sección superior se ve el código declarado en el

136 "renaissance of the collective author".

minuto 9:46 de la improvisación por el integrante josecaos, mientras que en la sección inferior de la ventana se ve la secuencia de líneas de código que fueron escritas por los integrantes josecaos, Emi y hernani entre los minutos 8:29 y 9:46.

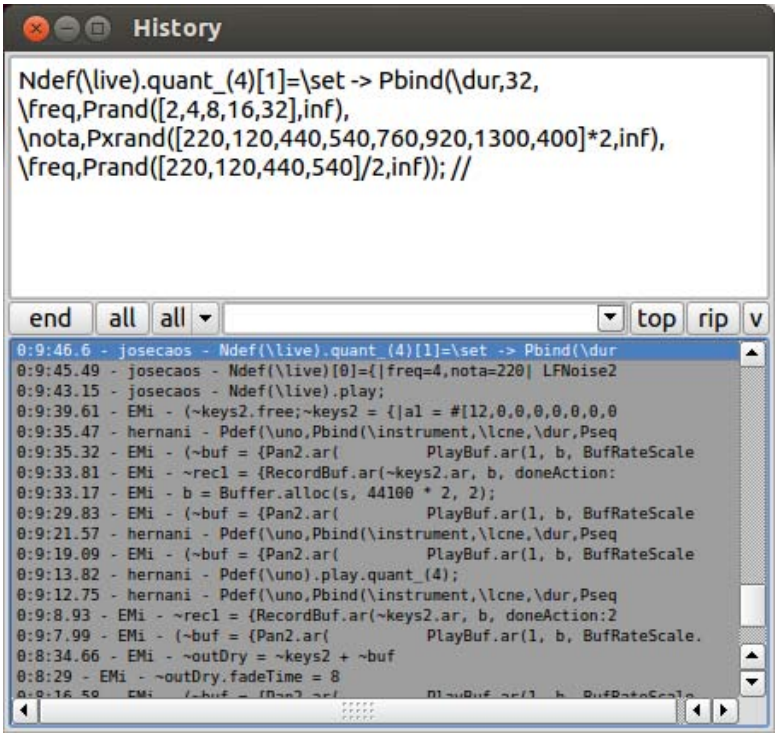


Figura 11. Ventana History del programa SuperCollider tomada de un concierto de LCNE. Toma de pantalla

La ventana History es el acceso a las líneas de código que cada integrante escribe y como cualquier texto digital escrito en una computadora puede copiarse y pegarse entre documentos. Sin embargo, en este caso no solo es una cuestión de copiar y pegar, se requiere además leer y analizar cada fragmento de código en un tiempo corto impuesto por la duración del concierto, pues aunque el código es compartido entre las computadoras de la red y está disponible para todos, utilizarlo resulta complicado si no hay una dinámica o

estrategia explícita para trabajar con él[137].

El código que recibimos de los demás integrantes durante un concierto se acumula en la pantalla lo que hace difícil su lectura y análisis. En este sentido, al referirse al networking social contemporáneo de la Web 2.0, Bazzichelli (2013) advierte acerca de un fallo en la comunicación debido a que ocurre de manera acelerada dificultando "la creación de un contexto conversacional profundo" (p. 134). En este sentido, Tiziana Terranova (2004) detecta un problema en la cultura de la información[138], la autora menciona que ésta no es solo contenido de comunicación que va de un emisor a un receptor, nos enfrenta a dinámicas informacionales expresadas de una manera más rápida de lo que la mente humana entiende. Por otro lado, Lee y Essl (2014) atribuyen al hecho de compartir código, durante una presentación de live coding en red, una carga cognitiva adicional:

> Podemos pensar que compartir el texto del código es bastante bueno, ya que tenemos acceso al código que puede reproducir los mismos objetos en una máquina local. Para un live coder, sin embargo, leer, interpretar y evaluar el fragmento de código son cargas cognitivas adicionales[139]. (Lee y Essl, 2014, párr. 6)

137 Por ejemplo, Rohrhuber et al. (2007) mencionan que la forma de trabajo que caracteriza a PowerBooks_UnPlugged es la de compartir instrumentos hechos con código, pequeños fragmentos de texto, a los que llaman *codelets*, de los que todos sus integrantes parten para hacerlos más complejos durante la improvisación, usando estrategias como llamada-respuesta, cadáver exquisito o algoritmos genéticos.

138 En el libro *Network Cultures: Politics for the Information Age,* Terranova plantea que el término información es una palabra que estamos acostumbrados a usar sin problematizar. Para hablar de este término la autora lo libera de dos prejuicios: pensar que "la información es 'el contenido de la comunicación'" (2004, p. 3) y que es "inmaterial" (ibíd.). Según la autora, esto ayuda a entender mejor las dinámicas informacionales. Para ella, la inmaterialidad se amplifica debido a desarrollos tecnológicos que hacen posible la "transmisión instantánea" y "distribución múltiple" (p. 6) de todo tipo de información como "imágenes, sonido, música, palabras, software, estadísticas, proyecciones, etc."(ibíd).

139 "One may think that sharing code text is good enough as one have access to the code that can reproduce the same objects in a local machine. For a live coder, however, reading, interpreting, and evaluating the code fragment are additional cognitive loads".

Entonces, si la aceleración y la carga cognitiva dificultan utilizar el código compartido en la red ¿por qué intercambiar y compartir código durante una presentación de live conding? A pesar de lo rápido que circula el código y la dificultad de utilizarlo, compartirlo es una referencia en el momento de improvisar, mantiene activo el intercambio de lo que sabemos hacer con él. En esta situación el código compartido está ahí, es algo que recuerda la presencia de los demás en la pantalla de las computadoras y la posibilidad de utilizarlo para modificarlo cuando sea posible. Lo importante de intercambiar código es que esta acción tiene la intención de compartir el conocimiento que está implícito en él.

3.3 La práctica de LCNE en el ensayo y concierto

La práctica de LiveCodeNet Ensamble es una forma de improvisación definida por el live coding en red en la que ver el código que escriben los demás y escuchar el sonido emergente son referentes que los integrantes del ensamble utilizan para crear música en conjunto. En esta práctica los integrantes interactúan a través de una red de computadoras local, intercambian y comparten código, se comunican por chat y sincronizan sus computadoras.

La práctica de LCNE sucede en dos momentos: el ensayo y el concierto. El ensayo es el espacio para la discusión y el tiempo para improvisar, intercambiar ideas, planear algún proyecto, ponerse de acuerdo antes de cada presentación o probar software en desarrollo[140].

La forma en la que improvisa LCNE es a partir de un plan general acordado en grupo durante los ensayos. Al respecto, Libertad comenta que la práctica del ensamble sucede a partir de una planeación previa en el ensayo que se ve influenciada por el contexto

140 Por ejemplo el software Tweetensamble que José Carlos Hasbun comenzó a desarrollar para realizar la residencia virtual de In-Sonora 9. Aunque no utilizamos este software para el propósito mencionado, algunos ensayos los dedicamos a probar la herramienta.

del concierto, lo que describe de la siguiente manera: “aunque sí lo platicamos, sí lo planeamos, sí hay toda una serie de cosas detrás, al final llegamos y tocamos de acuerdo a lo que hay en el lugar, o a lo que nosotros estemos, tal vez, sintiendo en ese momento” (L. Figueroa, comunicación personal, 27 de agosto 2015). Como se observa, Libertad no atribuye totalmente a la planeación la forma de llevar a cabo los conciertos, pues para ella lo que toca el ensamble se define en el momento del concierto. Esta dicotomía —entre planear e improvisar— está presente en ensayos y conciertos, por un lado hay una inclinación a la presentación planeada y por otro a la improvisación. El lugar donde se realiza el concierto y el público influyen en la forma en que improvisa el ensamble, a veces se decide cambiar lo que se ha planeado en el ensayo antes de comenzar a tocar.

En cuanto a planear una improvisación, Matthews (2012) opina lo contrario; este improvisador y compositor dice que la improvisación libre es aquella que se realiza “sin previo plan, estructura o acuerdo” (p. 19) y que:

> En realidad, lo que hace falta no es un acuerdo previo, sino la inteligencia, claridad de percepción y autodisciplina necesarias para poder asir, realmente, las implicaciones de la música en cada instante y actuar o reaccionar, seguir, proponer o incluso imponer el rumbo que vaya a tomar. (p. 61)

Lo que comenta Matthews no es posible sin conocimientos y experiencias previas, las cuales se acumulan a lo largo de la práctica que cada improvisador realiza. Si bien, Matthews se refiere la improvisación libre cuando habla de improvisación, el live coding en red opera de manera similar, aunque en este último caso se requiere un conocimiento especializado de un lenguaje de programación y la infraestructura que conecta las computadoras de los integrantes del ensamble. Para Emilio es importante realizar un

acuerdo previo pues de esto depende el diseño de los instrumentos[141] que empleará en cada concierto, lo que expresa de la siguiente manera: “como en acuerdos previos lo hacemos, o sea, hacia dónde vamos a ir y cómo le vamos a hacer; y a partir de eso yo ya voy generando mis instrumentos” (E. Ocelotl, comunicación personal, 20 de septiembre 2015).

Como se observa, los integrantes del ensamble desarrollan, diseñan y preparan sus materiales antes de los ensayos y conciertos. Es decir, una parte del código que emplean es escrita durante la exploración individual del lenguaje de programación. De esta manera, cada integrante cuenta con código escrito, el cual modificará con la intención de construir, complementar y transformar el sonido grupal durante cada ensayo y concierto. Ottavi (2008) menciona que a partir de la permisividad del software libre, los trabajos colectivos se caracterizan por la transformación, la modificación y la reapropiación de los materiales a los que se tiene acceso, el autor plantea que la conexión en red y la posibilidad de compartir agregan al trabajo de los creadores un “potencial de continuidad”[142] (p. 32), remplazando la noción de producto cuantificado por la de “formas indefinidas destinadas a desenvolverse de manera inesperada”[143] (ibíd.).

La idea de continuidad de Ottavi se ve reflejada en el trabajo que va desarrollando cada integrante a partir de las experiencias de tocar en red. Sin embargo, el trabajo de los integrantes de LCNE parte de la individualidad y más que modificar el código escrito por los demás, se modifica el propio en presencia de los otros. En este aspecto, quizá no hay una intervención directa sobre el código de otro integrante, pero éste es escrito en un contexto de creación colectiva. La estrategia de reescritura de código para crear música en el momento del ensamble propicia dicha continuidad. Desde esta perspectiva el código se escribe, crece y va acumulando conocimiento[144] y experiencia a lo largo de ensayos y conciertos. Además, el código va transformándose a partir de la interacción que realizan los

141 Por instrumentos Emilio se refiere al código que utiliza para improvisar, lo define como “el resultado de todas las inquietudes que he tenido durante el tiempo que me he acercado al código” (E. Ocelotl, comunicación personal, 20 de septiembre 2015).

142 “the potential to continue”.

143 “undefined forms destined to evolve unexpectedly”.

integrantes del ensamble en el momento que lo reescriben para improvisar. El código de cada integrante se transforma en la interacción con los demás y actualiza el sonido grupal emergente a partir de la combinación de los eventos sonoros que generan las computadoras.

Cada ensayo se extiende en el concierto, o en un nuevo ensayo, en los cuales continúan desarrollándose los procesos que han comenzado antes. En este sentido, la escritura de código es una práctica en progreso en la cual los integrantes del ensamble modifican el código fuente a partir de una interacción con el grupo y con su propio código. Esto se observa en la estrategia de escritura de código que describe José Carlos: "Trato de comenzar cada ensayo, posterior a un concierto, con algo distinto a lo anterior. Cada fin de ensayo guardo el archivo [de código] y cada nuevo ensayo retomo ese archivo […] para poder continuar desde donde me había quedado" (J.C. Hasbun, comunicación por correo electrónico, 23 de abril 2016).

A partir del comentario anterior, se puede argumentar que el código fuente que escribe cada integrante es un "mapa de conexiones en progreso" (Bazzichelli, 2013) (véase §2.3), pues éste ha sido escrito en varias etapas en las que los integrantes van imprimiendo parte de su experiencia, conocimiento e interacción con el grupo. Este código no funcionaría de manera individual fuera del contexto de la práctica en red, opera solo en el momento que los integrantes del ensamble se conectan para improvisar y realizar live coding. Retomando a Borgo (2006) (véase §2.4), el código ha sido escrito en influencia mutua durante cada improvisación que los integrantes realizan.

Los ensayos comienzan sin orden definido, cada integrante del ensamble se une a la improvisación conforme llega al lugar. En un concierto en cambio sucede de manera distinta, casi siempre los integrantes empiezan a improvisar al mismo tiempo. Libertad y Emilio coinciden que los conciertos comienzan cuando los integrantes llegan al lugar, realizan el montaje y las pruebas de sonido. Libertad añade que "en el caso de un concierto, sí nos ponemos de acuerdo para empezar a tocar […]. En algún momento decimos "bueno

144 Varela y Baca (2010) conciben el conocimiento en el contexto de lo que llaman sociedad del conocimiento como un potenciador de desarrollo en varias etapas: generación, gestión y divulgación.

ya", o nos dicen "bueno ya les toca" y entonces simplemente llegamos y nos ponemos a hacer sonar lo que traemos" (L. Figueroa, comunicación personal, UNAM, 23 de noviembre 2015). Por su parte Emilio comenta que las presentaciones comienzan ante la expectativa de lo que realizará el ensamble: "creo que a nosotros nos ayuda mucho esta dinámica del ritual del concierto, porque a veces tardamos mucho en empezar, pero la gente sabe que algo va a pasar" (E. Ocelotl, comunicación personal, 20 de septiembre de 2015). Emilio se refiere al tiempo que transcurre desde el momento que entramos al escenario y los primeros sonidos que comienzan a escucharse; tiempo en el que la audiencia puede observar que los integrantes están haciendo algo en sus computadoras, y si se proyecta el código, ve cómo aparecen las primeras líneas de código.

Una vez que se escuchan los primeros sonidos, la improvisación adquiere su rumbo. Emilio describe el desarrollo del concierto como una curva temporal "casi siempre, cuando tocamos, es como una curva así, hacia arriba y creo que a veces sí la logramos bajar y retomar, pero casi siempre es como una curva y más tendiente hacia el final es cuando baja, […] cuando vamos sacando los instrumentos, disminuyendo la densidad" (ibíd.). Para ilustrar la práctica de live coding en red de LCNE durante el concierto a continuación expongo la presentación realizada en la edición 7 del programa MOD Encuentros de Música Electrónica Viva.

3.4 Descripción de un concierto de LCNE

La práctica musical de LCNE se puede observar en el video[145] del concierto realizado el 22 de abril de 2015 en el Centro Cultural de España en México dentro del programa MOD

145 El video contiene la primer parte del concierto y puede verse en el siguiente enlace: <https://youtu.be/joo3kZEOef8>, cabe mencionar que el video fue proporcionado por Eduardo Obieta. El audio completo puede escucharse en <https://soundcloud.com/livecodenet-ensamble/lcne-en-vivo-mod7>

Encuentros de Música Electrónica Viva[146]. Analizar este concierto es interesante por los diferentes planos que presenta en los que se observa a los cinco integrantes del ensamble. El video muestra dos puntos de vista: el escenario y una pantalla arriba de los integrantes. A diferencia de las presentaciones de live coding, donde se exhibe ante al público la escritura de código fuente, en esta ocasión la proyección expone la actividad del ensamble en cuatro planos que cambian durante el concierto. Las imágenes de los diferentes planos proyectan la infraestructura tecnológica del ensamble —red de computadoras y equipo de sonido— y la interacción de los integrantes a través de esta infraestructura.

Como se observa en el video, cada integrante está concentrado en la pantalla y en el teclado de su computadora mientras escribe código o lo modifica. Se puede ver el gesto de teclear código, mover el ratón de la computadora y leer el código escrito. Se observa cómo interactúan los integrantes del ensamble a través de sus computadoras conectadas en red y del sonido grupal emergente que resulta de la improvisación. Como se menciona antes, la red de relaciones se construye en esta interacción.

Al centro de la mesa se observan los routers (minuto 1:48), los cuales emiten unas pequeñas luces que parpadean. Estos destellos lumínicos muestran la circulación de la información en la red de computadoras, el router revela la actividad del intercambio que mantienen los integrantes y la comunicación entre las computadoras. El intercambio de información ocurre en ese punto central y se concreta cuando llega a las pantallas de las computadoras de todos los integrantes. Esto se puede observar en el minuto 2:43 cuando aparece a cuadro una pantalla con diferentes ventanas en las que el integrante escribe, recibe y ve el código de los demás; también se puede ver la interfaz gráfica del reloj que lleva la sincronía.

146 LiveCodeNet Ensamble participó en el concierto MOD 7. El programa Encuentros de Música Viva MOD —organizado por Javier Lara y Mateo Lafontaine— plantea un formato en el que los invitados explican y muestran al público el proceso de la práctica musical que realizan mediante una plática y un concierto en el que varias cámaras muestran, en la pantalla del lugar, la actividad de los músicos en el momento de realizar el concierto <http://ccemx.org/mod-7>

Figura 12. LiveCodeNet Ensamble. 22 de abril de 2015, Centro Cultural de España, Ciudad de México. Fuente: Archivo fotográfico de Eduardo H. Obieta.

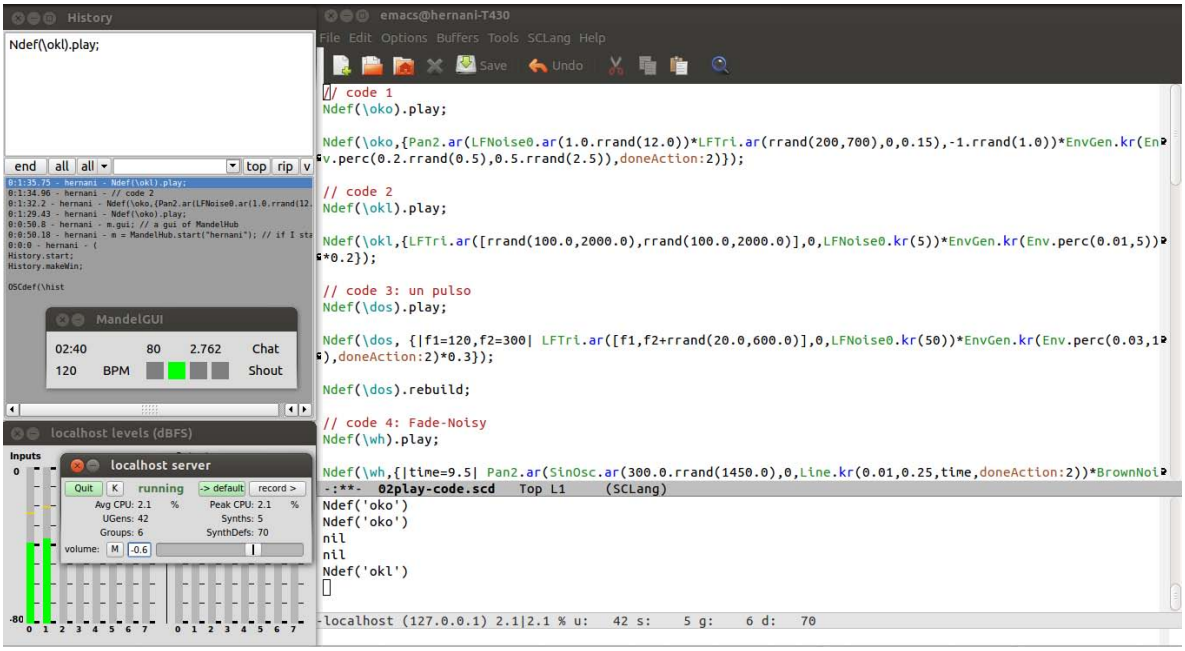


Figura 13. Pantalla para improvisar con LCNE. Toma de pantalla

El ejemplo de la figura 13 ilustra a detalle la pantalla de mi computadora. En la gráfica se puede ver, del lado derecho, la ventana donde se escribe el código cuya parte inferior muestra una serie de avisos, errores en el código o afirmaciones de que todo ha salido bien; de lado izquierdo hay diferentes ventanas: arriba se encuentra la ventana History donde se concentra y visualiza el código de todos los integrantes de manera cronológica; más abajo está la ventana Mandel GUI que indica el tiempo y estado de la sincronización, así mismo los niveles de entrada y salida y la interfaz del servidor de SuperCollider.

El ejemplo de la figura 14 muestra un fragmento de la secuencia de código escrito por tres integrantes del ensamble, extraído del documento que genera la ventana History. El texto del ejemplo expone algunas líneas de código como se escriben durante una presentación y la forma en que quedan guardados en un documento generado por la

extensión History. Este fragmento de código fue escrito durante un concierto distinto[147] al ejemplo del video, sin embargo, la secuencia ocurre de manera similar en cada presentación.

```
 // - 0:13:4.42 - (hernani)

Pdef(\uno,Pbind(\instrument,\lcne,\dur,Pseq([0.25],inf),\frec,Pseq([80,8
0,120],
 inf),\rf,Pseq([80],inf),\pulse,Pseq([0.1,0.5,0.4],inf),\rel,
 Pseq([0.1,0.01,0.5]),\vol,0.25));

 // - 0:13:5.59 - (EMi)
 (
 ~buf = {Pan2.ar(
      PlayBuf.ar(1, b, BufRateScale.kr(b) * [-1,-0.5],
            Impulse.kr(~tempo*0.5),
            BufFrames.kr(b) * 0.4)).fold2(0.1) * 0.75};
 )

 // - 0:13:13.78 - (josecaos)
 Ndef(\live).quant_(4)[1]=\set ->

Pbind(\dur,0.25,\freq,Prand([2,4,8,16,32],inf),\nota,Pxrand([220,120,440
,540,
 760,920,1300,400]/2,inf),\freq,Prand([220,120,440,540]/2,inf)); //
```

Figura 14. Extracto de la historia de una improvisación con código de LiveCodeNet Ensamble

Aunque no se puede probar, con el texto del ejemplo, que el código es resultado de la interacción entre los integrantes, lo que quiero mostrar con él es que las líneas de código

147 Concierto realizado el 24 de septiembre 2015 en la Antigua Academia de San Carlos de la Ciudad de México, en el marco de la sesión 44 de Live Cinema y Arte Sonoro. El sonido de este concierto se puede escuchar en <https://soundcloud.com/livecodenet-ensamble/lcne-antigua-academia-de-san-carlos>

expuestas son resultado de un proceso de escritura en progreso que los integrantes realizan en el momento teniendo como referencia el código que escriben los demás integrantes y el sonido que se genera, es decir, este código solo pude ser escrito de esta manera en la interacción con los otros integrantes durante una improvisación. Cabe mencionar que el código es una referencia tanto visual, por su lectura, como auditiva, por la escucha del sonido que resulta.

Para concluir este capítulo, se puede decir que compartir e intercambiar código fuente en la práctica de LCNE es posible debido a la infraestructura tecnológica y configuración de la red de computadoras que utilizan. Esta configuración determina la forma en que improvisa el ensamble; a su vez, la práctica de live coding realizada determina la configuración de la red tecnológica. Red de computadoras y red de relaciones coexisten, en términos de Sorensen (2007), como una red o ensamblaje socio-material.

Compartir e intercambiar en el momento de realizar live coding en red es una decisión individual gestionada por el software de la red. Aunque existe esta posibilidad, parece que la estrategia de trabajar con el código de otro integrante no es preponderante en las dinámicas del ensamble, ya que leer, analizar y utilizar el código fuente compartido ocurre en un contexto en que la información generada es mucha, ocurre rápido y en poco tiempo. En este sentido, no hay una dinámica grupal muy clara que guíe el resultado sonoro de la improvisación a partir de intercambiar y compartir código.

Entonces, ¿por qué compartir e intercambiar código si no tiene una injerencia directa en el sonido de la improvisación? En el caso del live coding en red es un gesto que se expresa en el dejar ver. El código que se muestra a los demás es un referente de lo que sabemos hacer en el campo del live coding, es un paso en la organización para tocar juntos y realizar una práctica artística, implica exponer y confrontar el pensamiento individual implícito en el código ante los demás.

Ante la insistencia en el dejar ver, habría que hacerlo también en la escucha. Por un lado, LCNE usa una red abierta de computadoras que permite ver el código de todos, por

el otro, un sistema de sonido que canaliza las señales de audio hacia un punto. Esto provoca que los integrantes negocien el sonido colectivo en una retroalimentación compleja, a través de la escucha grupal y de la combinación sonora que generan las acciones de todos los integrantes.

Esta forma de gestionar el código y el sonido genera una tensión que define la forma de crear música del ensamble, la cual parece que está encaminada más por una interacción a través de la escucha que de la lectura de código compartido. Es en esta forma de interacción que la red de relaciones se construye y el código fuente se convierte en una especie de mapa en el que van quedando inscritos conocimientos, interacciones y experiencias. En este sentido, es posible decir que el live coding en red es más parecido a la improvisación libre que a una dinámica de intercambio de código fuente, pues opera, en mayor medida, con base en una conversación sonora.

Conclusiones

Este trabajo ha estudiado el live coding en red a través de una serie de asociaciones que lo conectan con el concepto de networking. A su vez, el caso de LiveCodeNet Ensamble ha servido para mostrar cómo sucede esta práctica desde un análisis en el marco de las asociaciones mencionadas. A continuación expongo los resultados obtenidos a partir de los objetivos y preguntas de investigación, las conclusiones del trabajo y las posibles líneas de investigación a futuro.

Al analizar los aspectos técnicos del lenguaje de programación y la red de computadoras, con base en lo referido por diferentes autores, encontré que el código fuente son instrucciones escritas que, dirigidas a la computadora, producen acciones. En el caso del live coding musical, dichas acciones generan sonido. Los lenguajes interpretados y de alto nivel permiten escribir código fuente y modificarlo sin detener el funcionamiento de un programa, esta característica permite interactuar de manera escrita con el sonido, a través del código fuente, lo que posibilita el desarrollo de una práctica de improvisación, ya que resulta factible crear música en el momento. En este contexto, el live coder o practicante de live coding se define en una dicotomía de músico/programador, quien se expresa a través de la escritura de código fuente durante una improvisación musical.

Una red de computadoras está diseñada para compartir recursos e intercambiar información; así mismo, para transferir datos a la distancia. Estas características son aprovechadas en el campo de la música por computadora para crear música de manera conjunta. En el live coding en red se refleja en la posibilidad de realizar una práctica de creación musical grupal en la que dicha infraestructura permite intercambiar código fuente, compartir recursos, comunicar y sincronizar las computadoras. A su vez, los discursos de la ética hacker, el software libre y el open source, que hacen énfasis en los principios de compartir e intercambiar conocimiento y abrir el código fuente de un programa, han permeado el live coding en red, lo que se observa en su dinámica de mostrar el código.

Al establecer el vínculo entre el live coding en red y las prácticas de networking encontré que esta asociación puede trazarse a partir de una serie de conexiones con diferentes prácticas. La primera es considerar al live coding en red como un caso de improvisación colectiva, pues desde ahí se plantea como una práctica en progreso que explora la creación de redes de relaciones entre los live coders en el momento de construir el sonido grupal durante un concierto o ensayo. La segunda conexión, con un origen en los años setenta, se da entre la música en red y el movimiento hacker a partir de sus discursos de acceso, apertura, intercambio y compartición a través de la interconexión. En el primer caso desde la perspectiva de una práctica de creación musical colectiva, en el segundo desde la búsqueda de nuevas formas de conexión entre personas a través de las computadoras. En este sentido, el live coding en red, como práctica de improvisación colectiva y música en red, conecta ambas prácticas a partir del lenguaje abierto de la improvisación libre y la apertura para compartir e intercambiar datos en una red de computadoras.

En cuanto al análisis y descripción de la práctica de LCNE, bajo los aspectos técnicos y sociales analizados, es posible decir que las características de la red de computadoras de LCNE —local, en forma de estrella, par a par y de difusión— posibilitan una práctica en la que los integrantes tocan en el mismo lugar con un diseño abierto que permite la circulación de datos entre las computadoras del grupo. Por otro lado, el sonido que genera el ensamble es gestionado por un sistema de sonido configurado en estéreo que canaliza las señales de audio a un punto en el cual se combinan para obtener el resultado sonoro del grupo. A partir de lo antes mencionado, es posible plantear que la interacción entre los integrantes del ensamble ocurre en dos puntos que retroalimentan la improvisación: en la red de computadoras y en la salida del sistema de sonido. En el primero caso a partir de ver el código fuente en las pantallas de las computadoras, en el segundo de escuchar el sonido resultante en las bocinas.

En cuanto al concepto de networking (Bazzichelli, 2008; 2013), éste puede plantearse en relación a la práctica de live coding en red de LCNE con base en las tres

formas que la autora define el término: primero, la red de relaciones, en la práctica del ensamble, se crea en el momento de construir el sonido grupal de la improvisación en un diálogo sonoro generado por el código que escribe cada integrante del ensamble y a través de la escucha. Segundo, la estrategia de compartir e intercambiar es posible llevarla a cabo por la configuración de la red de computadoras que está diseñada para la circulación abierta de datos, y por los acuerdos que definen la práctica del ensamble. Tercero, el mapa de conexiones en progreso está inscrito en el código fuente que escribe cada integrate. Este último planteamiento es posible si consideramos al código no solo como instrucciones para generar sonido sino como contenedor de conocimientos, experiencias e interacciones que se acumulan en cada ensayo y concierto.

En cuanto a las preguntas de investigación, éstas pueden responderse de la siguiente manera: el live coding en red es una práctica colectiva de música con computadoras interconectadas en la que se crea una red de relaciones en el momento de construir el sonido grupal de una improvisación. El trabajo individual que cada integrante realiza en el grupo, al escribir y modificar código fuente para generar sonido, articula la música durante ensayos y conciertos en las acciones de mostrar, compartir e intercambiar código fuente, así como en la de escuchar el resultado sonoro. La red de computadoras de LCNE determina las dinámicas con las que el ensamble realiza live coding en red y posibilita el intercambio y la compartición de datos entre las computadoras de los integrantes así como la sincronización de sus computadoras; esto, a su vez, permite compartir conocimiento, ideas y experiencias que se inscriben en el código fuente.

Para concluir este trabajo es posible decir que las nociones de compartir e intercambiar establecen un vínculo entre la red de computadoras y la red de relaciones en la práctica de live coding en red, en el primer caso como funciones de la red de computadoras, en el segundo como estrategia que permite la circulación de ideas, experiencias y conocimientos. Las redes de computadoras, al estar diseñadas para compartir recursos e intercambiar información y datos, presentan una apertura que invita a utilizar el código que realizan los demás como estrategia de creación musical colectiva durante una

improvisación. Sin embargo, dinámicas de este tipo requieren una estrategia explícita previamente formulada, por lo que es necesario crear reglas y directrices que activen esta dinámica durante los conciertos que se ven delimitados por el tiempo que duran, la velocidad con que circulan los datos en la red y la carga cognitiva que supone escribir, leer y escuchar de manera individual y en grupo.

La forma en la que está configurado el sistema de sonido de LCNE establece un campo en el cual sus integrantes negocian la construcción del sonido grupal a partir de la combinación de los eventos sonoros que generan. Escuchar el resultado y modificar el código fuente permite a los integrantes cambiar el transcurso del sonido colectivo lo cual sucede en influencia mutua y en una transformación continua de los eventos sonoros en el momento de improvisar. Es en la interacción para construir el sonido en grupo que se crea la red de relaciones del ensamble en el live coding en red.

Algo que no queda muy claro es cómo afectan la compartición y el intercambio al resultado sonoro y a las dinámicas que se dan en la improvisación grupal. Al respecto, es posible inferir, a partir de las entrevistas realizadas, que en una presentación de live coding en red los integrantes de LCNE interactúan en mayor medida guiados por la escucha del sonido resultante que por el intercambio de código fuente. Por lo tanto se puede plantear que el live coding en red, en el caso del ensamble, se desarrolla de manera más cercana a la improvisación libre que a un trabajo con base en el intercambio y compartición de código fuente. Sin embargo, a partir de esta interrelación —escribir código y escuchar su resultado — se configura la práctica de live coding en red.

En cuanto a futuras líneas de investigación, es posible ampliar la perspectiva técnica y social que supone la relación propuesta en este trabajo hacia un análisis de las implicaciones estéticas, culturales, políticas y económicas del live coding en red en un contexto tanto local como global. Por otro lado, queda pendiente problematizar y analizar el caso del live coding en red a la distancia y, con mayor profundidad, las implicaciones de la escucha y la colaboración en el caso del live coding en red. Pero es desde la siguiente propuesta, en la que este trabajo ha agotado su investigación, que puede retomarse su

estudio: en el intercambio y la compartición de información que sucede en una red de computadoras hay también uno de ideas, experiencias y conocimiento que tiene implicaciones en la forma de creación musical. A partir de lo anterior surge la siguiente pregunta ¿de qué manera el intercambio y la compartición de información y conocimiento determinan la práctica musical del live coding en red?

Por último, quisiera resaltar el potencial del live coding en red como práctica artística que devela el papel social de la música en un contexto tecnológico, lo que ofrece la oportunidad de juntarnos y organizarnos para crear de manera conjunta en una actividad que permite emerger nuestra personalidad dentro de lo colectivo para dialogar y negociar, a través del sonido y el código fuente, en una configuración que va más allá de la topología de la red y la interconexión.

Bibliografía

Alonso, Ch. (2008). *Improvisación libre. La composición en movimiento*. Baiona: Dos Acordes.

Arns, I. (2005). El código como acto de habla performativo. *Artnodes*, 4. doi: http://doi.org/10.7238/a.voi4.727

Bailey, D. (2010). *La improvisación: Su naturaleza y su práctica en la música*. Gijón: Ediciones Trea.

Barceló Ordinas, J.M., Íñigo Griera, J., Martí Escalé, R., Peig Olivé, E. y Perramon Tornil, X. (2004). *Redes de computadores*. Barcelona: Universitat Oberta de Catalunya.

Bazzichelli, T. (2008). *Networking: The Net as Artwork.* Aarhus: Digital Aesthetics Research Center.

Bazzichelli, T. (2013). *Networked Disruption: Rethinking Oppositions in Art, Hacktivism and the Business of Social Networking*. Aaruhs: Digital Aesthetics Research Center.

Bell, R. (2014). Considering Interaction in Live Coding through a Pragmatic Aesthetic Theory. e*Contact!*, 16(2). Recuperado de http://econtact.ca/16_2/bell_livecoding.html

Bell, S. (s.f.). Live coding brings programming to live – an interview with Alex McLean [Entrada de blog]. *British Science Association blog*. Recuperado de http://www.britishscienceassociation.org/blog/live-coding-brings-programming-to-life-an-interview-with-alex-mac

Berenguer, J. M. (2007). ¿Tiempo real? *Artes, La Revista*, 7(13), 37-43.

Blackwell, A., McLean, A., Noble, J., y Rohrhuber, J. (2013). *Collaboration and leraning through live coding*. (Reporte del Seminario Dagstuhl 13382). Schloss Dagstuhl: Leibniz-Zentrum für Informatik.

Booth, G. y Gurevich, M. (Mayo, 2012). *Collaborative composition and socially constructed instruments: Ensemble laptop performance through the lens of ethnography.* Documento presentado en 12th International Conference on New Interfaces for Musical Expression, Ann Arbor, Michigan.

Borgo, D. (2006). Sync or Swarm: Musical Improvisation and the Complex Dynamics of Group Creativity. En Futatsugi, K., Jouannaud, J.P. y Meseguer, J. (Eds.), *Algebra, Meaning and Computation: Essays dedicated to Joseph A. Goguen on the Ocassion of His 65th Birthday* (pp. 1-24). Berlin: Springer-Verlag.

Brookshear, G. (2012). *Introducción a la computación* (11a ed.). Madrid: Pearson Educación.

Brown, Andrew R. (2016). Performing with the other: the relationship of musician and the machine in live coding. *International Journal of Performance Arts and Digital Media*, 12(2), 179-186. doi: http://dx.doi.org/10.1080/14794713.2016.1227505

Brown, Ch. y Bischoff, J. (2002). Indigenous to the net: Early Network Music Bands in the San Francisco Bay Area [Artículo en línea]. Recuperado de http://crossfade.walkerart.org/brownbischoff/index.html

Chandler, A. y Neumark, N. (Ed.). (2005). *At a Distance: Percursors to Art and Activism on the Internet*. Cambridge: The MIT Press.

Chiantore, L., Domínguez, A. y Martínez, S. (2016). *Escribir sobre música*. Valencia: Musikeon.

Chun, W. (2008). On "Sourcery," or Code as Fetish. *Configurations*, 16(3), 299-324. doi: 10.1353/con.0.0064

Collins, N., Cottle, D., y Wilson, S. (Ed.). (2011). *The SuperCollider Book*. Londres: MIT Press.

Collins, N., McLean, A., Rohrhuber, J. y Ward, A. (2003). Live coding in laptop performance. *Organised Sound*, 8(3), 321-329.

Cox, G. (2015). What does Live Coding know? En McLean, A., Magnusson, T., Ng, K., Knotts, S., Armitage, J. (Eds.), *International Conference on Live Coding* (pp. 93-97). Leeds.

Cox, G. y Mclean, A. (2012). *Speaking Code: Coding as Aesthetic and Political Expression*. Cambridge: MIT Press.

Cramer, F. (2005). *Words Made Flesh: Code, Culture, Imagination*. Rotterdam: Piet Zwart Institute.

Di Próspero, C. (2015). Live coding. Arte computacional en proceso. Contenido. *Arte, Cultura y Ciencias Sociales*, (5), 44-62.

DiBona, Ch., Ockman, S. y Stone, M. (Eds.). (1999). *Opensources: Voices from the Open Source Revolution*.

Fuller, M. (Ed.). (2008). *Software Studies: A lexicon*. Cambridge: The MIT Press.

Galloway, A. (2004). *Protocol: How control exists after decentralization*. Cambridge: MIT Press.

Himanen, P. (2001). *The Hacker Ethic and the Spirit of the Information Age*. New York: Random House Trade Paperbacks.

Herrera, M. (2013). *El gesto en control del gesto: Diseño e implementación de un framework para la interacción gestual natural* (Tesis de maestría). Recuperado de Tesiunam Tesis del Sistema bibliotecario de la UNAM.

Kelly, C. (2009). *Cracked Media: The sound of Malfunction*. Cambirdge: Mit Press.

Knotts, S. (2015). Changing Music's Constitution: Network Music and Radical Democratization. *Leonardo Music Journal*, 25, 47-52.

Knotts, S. y Collins, N. (2014, junio). *The Politics of Laptop Ensembles: A Survey of 160 Laptop Ensembles and their Organisational Structures.* Documento presentado en NIME 14, Londres.

Lee, S. W. y Essl, G. (2014, septiembre). *Models and Opportunities for Networked Live Coding*. Documento presentado en Live Coding and Collaboration Symposium, Birmingham.

Lee, S. W. y Essl, G. (2015, julio). *Live Writing: Web Based Text Editor for Asynchronous Playback of Live Coding and Wirting*. Documento presentado en International Conference on Live Coding 2015, Leeds.

Lee, S. W., Freeman, J., Colella, A., Yao, S. y Van Toyer, A. (2012, abril). *Evaluating Collaborative Laptop Improvisation with LOLC*. Documento presentado en Symposium on Laptop Ensembles and Orchestras, Baton Rouge.

Levy, S. (2010). *Hackers. Heroes of the Computer Revolution*. Sebastopol: O'Reilly.

Martin, Ch. y Gardner, H. (2015). That Sync Feeling: Networked Strategies for Enabling Ensemble Creativity in iPads Musicians. En Gifford, T. (Ed.), *CreateWorld 2015: A Digital Arts Conference* (pp. 35-40), Brisbane: Griffith University.

Matthews, W. (2012). *Improvisando. La libre creación musical*. Madrid: Turner Música.

Mattin (2009). Volverse Frágil. En Mattin y Iles, A. (Eds.), *Ruido y Capitalismo* (pp. 21-25). San Sebastián: Arteleku.

McCartney, J. (2003, septiembre). *A Few Quick Notes on Opportunities and Pitfalls of the Application of Computers in Art and Music*. Documento presentado en Ars Electronica 2003, Linz.

McKinney, C. y McKinney, Ch. (2012, septiembre). *OSCthulhu: Appliyng Video Game Stated-Based Synchronization to Network Computer Music*. Documento presentado en ICMC 2012, Institue for Sonic Arts Research, Liubliana.

McLean, A. (2011). *Artist-Programmers and Programming Languages for the Arts*. (Tesis de Doctorado). Goldsmiths, University of London, Londres.

Moraes Costa, R. L. (2015). A improvisação livre, a construção do som e a utilização das novas tecnologias. *Revista Música Hodie*, 15(1), 119-131.

Norton, P. (2014). *Introducción a la computación* (6a ed.). Ciudad de México: McGraw-Hill Interamericana.

Ogborn, D. (2016). Live coding together: Three potentials of collective live coding. *Journal of Music, Technology and Education,* (9)1, 17-31. doi: 10.1386/jmte.9.1.17_1

O'Regan, G. (2013). *Giants of Computing: A Compendium of Select, Pivotal Pionners*. Londres: Springer-Verlag.

Ottavi, J. (2008). The 'Free' and New Creative Practices: Open Source Modular Art-efacts. En Mansoux, A. y De Valk, M. (Eds.), *FLOSS + Art* (pp. 28-33). Poitiers: GOTO10.

Prada, J.M. (2012). *Prácticas Artísticas e Internet en la Época de las Redes Sociales*. Madrid: Ediciones Akal.

Roberts et al. (2015, julio) *Sharing Time and Code in a Browser-Based Live Coding Environment*. Documento presentado en International Conference on Live Coding 2015, Leeds.

Rohrhuber, J. (2007). Network Music. En Collins, N. y D'Escrivan, J. (Eds.), *The Cambridge Companion to Electronic Music* (pp. 140-155). Cambridge: Cambridge University Press.

Rohrhuber, J. y de Campo, A. (2009). Improvising Formalsation: Conversational Programming and Live Coding. En Assayag, G., y Gerzso, A. (Eds.). *New Computational Paradigms for Computer Music*. Paris: Delatour France.

Rohrhuber, J., de Campo, A., Wieser, R., van Kampen, J., Ho, E. y Hölzl, H. (2007, diciembre). *Purloined Letters and Distributed Persons*. Documento presentado en Music in the Global Village Conference, Mücsarnok, Budapest.

Sorensen, E. (2007). The time of materiality. *Forum: Qualitative Social Research*, 8(1). doi: http://dx.doi.org/10.17169/fqs-8.1.207

Tanenbaum, A. y Wetherall, D. (2012). *Redes de computadoras* (5ª ed.). Naucalpan de Juárez: Pearson.

Terranova, T. (2004). *Network Culture. Politics for the Information Age*. London: Pluto Press.

Varela Salas, L. y Baca Pumarejo, J. R. (2010). La gestión del conocimiento. *Contribuciones a las Ciencias Sociales*, 10. Recuperado de http://www.eumed.net/rev/cccss/10/vsbp.htm

Wang, G. (2007). A history of programming and music. En Collins, N. y D'Escrivan, J. (Eds.), *The Cambridge Companion to Electronic Music* (pp. 55-71). Cambridge: Cambridge University Press.

Wang, G. y Cook, P. (2004, junio). *On-the-fly Programming: Using Code as an Experessive Musical Instrument*. Documento presentado en NIME 2004, Hamamatsu.

Wang, G., Mirsa, A., Davidson, P. y Cook, P. R. (2005, septiembre). *Co-Audicle: A Collaborative Audio Programming Space.* Documento presentado en ICMC 2005, Barcelona.

Ward, A., Rohrhuber, J., Olofsson, F., McLean, A., Griffiths, D., Collins, N., y Alexander, A. (2004). Live algorithm programming and a temporary organization for its promotion. En Gourinova, O. y Shulgin, A. (Eds.). *Read_Me: Software Art & Cultures.* Aarhus: Aarhus University Press.

Weinberg, G. (2005). Interconnected Musical Networks: Toward a Theoretical Framework. *Computer Musica Journal*, 29(2), 23-39.

Wilson, S., Lorway, N., Coull, R., Vasilakos, K. y Moyers, T. (2014). Free as in BEER: Some Explorations into Structured Improvisation Using Networked Live-Coding Systems. *Computer Music Journal*, 31(1), 54-64. doi:10.1162/COMJ_a_00229

Zmölnig, J. y Eckel, G. (2007, agosto). *Live coding: an overview*. Documento presentado en ICMC 2007, Copenhague.

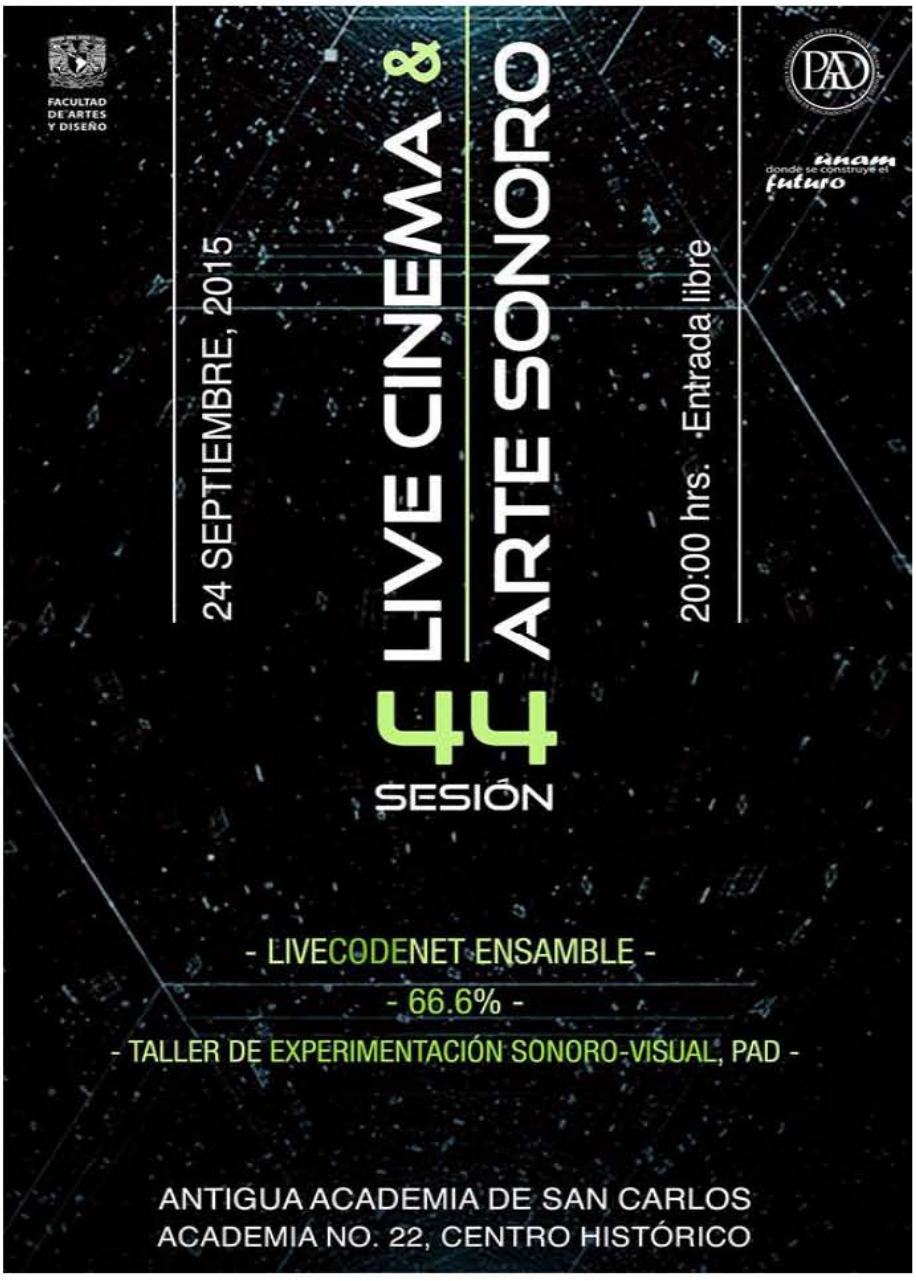
Anexos

Anexo A. Código de conexión

```
//=============================================
//1 LiveCodeNet Ensamble código de inicio
//=============================================
s.boot;
s.meter;
s.makeGui;

//=============================================
//2 Conectar al Router
//=============================================
(
q = ();
NetAddr.broadcastFlag = true;
q.addrs = (0..7).collect { |x|
            NetAddr("255.255.255.255", 57120 + x)
};
q.sendAll = { |q ... args|
            q.addrs.do { |addr|
                        addr.sendMsg(*args)
            }; ""
}
);

//=============================================
//3 History - para compartir código con los demás
//=============================================
(
History.start;
History.makeWin;
```

```
OSCdef(\hist, { |msg|
      /*msg.postln;*/
      History.enter(msg[2].asString,msg[1]);
}, \hist).fix;
History.localOff;
History.forwardFunc = { |code|
      q.sendAll(\hist, \hernani, code);
};
);

//=============================================================
//4 (opcional si se quiere tocar en sincronía
//Requiere BenoitLib: https://github.com/cappelnord/BenoitLib
//=============================================================
m = MandelHub.start("hernani"); // si empiezo como líder
m.gui; // a gui of MandelHub

MandelHub.join("hernani", action:{m=MandelHub.instance});// para
seguir al líder
m.gui; // Interfaz de usuario MandelHub

m.takeLead; // para tomar el liderazgo

m.timer.reset; // para restablecer el tiempo

m.changeTempo(122/60, 1); // para cambiar el tempo

m.clear; // Finaliza la conexión de MandelGUI
```

Anexo B. Carteles de conciertos



FACULTAD DE MÚSICA – FACULTAD DE ARTES Y DISEÑO

LIVE CINEMA & ARTE SONORO

LiveCodeNet Ensamble
Interconexión y colaboración algorítmica experimental.
Libertad Figueroa, Emilio Ocelotl, Eduardo H Obieta, José Carlos Hasbun y Hernani Villaseñor.

TOME-UACM
Taller Orquesta de Música Experimental de la Universidad de la Ciudad de México
Conducción Ramsés Luna

EXPANDED CINEMA
Dirección Manuel Trujillo Morris

Taller de Experimentación Sonoro-visual en tiempo real, PAD, UNAM
Luisa Fernánda Gutiérrez, Diana Fernández, José Ricardo Guzmán, Cuitláhuac Oropeza, Pedro Alayón, Edgar Olvera Yerena

SESIÓN 36

14 MAYO
19:30 hrs.
SALÓN A–10

Facultad de Música// Xicoténcatl # 126, Col. Del Carmen, Coyoacán.

FACULTAD DE ARTES Y DISEÑO

unam donde se construye el futuro