Enfrentando la interfaz, nuevas perspectivas en las artes vivas

Esteban Betancur Gutíerrez

Instituto Tecnológico Metropolitano
Facultad de Artes y Humanidades
Medellín, Colombia
2017

Enfrentando la interfaz, nuevas perspectivas en las artes vivas

Esteban Betancur Gutíerrez

Tesis de grado presentado como requisito parcial para optar al título de:
Magister en Artes Digitales

Trabajo dirigido por:

Gloria Mercedes Díaz, PhD.
Departamento de Sistemas de Información, Facultad de Ingenierías
Instituto Tecnológico Metropolitano

Y

Wilson Javier Sarmiento, PhD.
Facultad de Ingeniería
Universidad Militar Nueva Granada

Grupo de Investigación:
Grupo Artes y Humanidades

Instituto Tecnológico Metropolitano
Facultad de Artes y Humanidades
Medellín, Colombia
2017

Al Amor y a quienes me lo han mostrado

Agradecimientos

Dra. Gloria M. Diaz Cabrera
Dr. Wilson J. Sarmiento
Dr. David Ogborn

Algo0ritmos:
Federico Lopez, Santiago Benitez, Andres Uribe, Viviana Ramírez, Juan Jaramillo, David Crowley, Danny Zurc

Cinevivo:
Luis Castro - el malo - , Luis Rodriguez - el bueno - ,Erica Florez, Sara Henao, Edwin Cortes, Tatiana Duque, Catherine Restrepo, Raul Benito

Acorde:
Miguel Vargas, Diego Molina, Santiago Restrepo

Cybernetic Orchestra:
Jamie Beverly, Luis Navarro del Angel

CICLUX

Al ITM (profesores y administrativos) en especial, a J.Diego, Elena y Santiago, Compañeros de Maestría. Parque Explora (Colaboratorio) Camilo Cantor. Platohedro (a todos en la casa).
A los live coders del mundo por ayudar creando y respondiendo.
Y a los que siempre estan ahí:
Toda la familia del alma, Pastor, Cruz, Cami, y mis chiquitos Lau y Fua Ma.

Resumen

Este trabajo muestra la experiencia de diseño, desde la concepción, pasando por la depuración y final implementación de las interfaces usadas por artistas, músicos y entusiastas de varios colectivos artísticos de la ciudad de Medellín durante el periodo comprendido entre 2013 a 2017; Los colectivos con los que se trabajó fueron: Colectivo CICLUX, Semillero de investigación-creación CINEVIVO, Semillero de investigación-creación ACORDE, Colectivo Algo0ritmos. También, la Cybernetic Orchestra de la Universidad McMaster en Hamilton, Ontario, Canadá.

La tesis centra sus esfuerzos en el trabajo "en vivo" y especialmente en la producción audiovisual e intenta analizar desde varias perspectivas, la evolución de cada interfaz, desde los prototipos hasta los diseños finales y cómo estos cambios fueron afectando la evolución de la obra producto del trabajo con la interfaz bien sea esta una interfaz gráfica, física o basado en la idea del código de programación computacional como interfaz para el trabajo en tiempo real.

Los capítulos podrán verse de forma independiente y no es necesaria una lectura lineal.

Palabras clave: Interfaz, Live Coding, Live Cinema, Diseño Centrado en el Usuario, DIY, Chuck, Chmusick, CQenze, CineVivo.

Abstract

This work shows the design experience, from conception to debugging and the final implementation of the interfaces used by artists, musicians and enthusiasts of various artistic groups in the city of Medellín during the period from 2013 to 2017; groups with which we worked were: Colectivo CICLUX, Semillero de investigación-creación CineVivo, Semillero de investigación-creación ACORDE, Colectivo Algo0ritmos. Also the Cybernetic Orchestra of McMaster University in Hamilton, Ontario, Canada.

This thesis focuses on live works and especially on audiovisual production and tries to analyze from various perspectives the evolution of each interface, from prototypes to final designs and how these changes were affecting the evolution of the artwork as a product of the work with the interface, either it is a graphical interface, physical or based on the idea of computational programming as interface for the work in real time.

Chapters can be viewed independently and no linear reading is required.

Contenido

Lista de símbolos

Abreviaturas

Abreviatura	Término
GUI	Graphical User Interface
DSL	Domain Specific Languages
$MIDI$	Musical Instrument Digital Interface
OSC	Open Sound Control
OF	Open Frameworks
UDP	User Datagram Protocol
RPI	Raspberry PI
HCI	Human Computer Interaction
EDM	Electronic Dance Music
DIY	Do It Yourself
OOP	Object Oriented Programming
$JITLIB$	Just In Time Library
SC	SuperCollider
PD	PureData

1. Introducción y contexto

1.1. Obertura

Esta tesis documenta el proceso de desarrollo de interfaces generadas a través de un esfuerzo colaborativo, para la producción de trabajos audiovisuales, e investiga la evolución de estas interfaces. ¿La razón? los cambios y las transformaciones a las que llevó la obra, que era su objetivo, y cómo en reacción a estos cambios hubo transformaciones en la mente de los artistas, vistas desde: el cambio de perspectivas y nuevas ideas en el proceso de diseño.

Dado que las interfaces que se describirán a lo largo del texto fueron producto de varias mentes y varias disciplinas, es necesario hablar del proceso de conceptualización y puesta en común de las diferentes ideas que fueron usadas en la construcción y depuración de cada una de ellas, y las obras resultado. Así mismo, es necesario mostrar este proceso como una transformación constante, un aceptar y discutir, no sólo con las ideas externas sino con el curso de los cambios internos y la evolución de la obra de la mano de este discurrir. Este concepto se englobará en uno más general que lo fundamenta: la violencia y el error como factores de cambio.

Uno de los objetivos de fondo es mostrar cómo la producción audiovisual en vivo, junto a la cada vez más evidente capacidad de cómputo en tiempo real de grandes cantidades de información, liberada del tradicional miedo al error en la ejecución artística, es una gran herramienta para el desarrollo de interfaces que potencian las relaciones humano-máquina (máquina algorítmica, no necesariamente un computador en el sentido común del termino, y extendiendo este concepto a personas como ejecutantes), en sistemas para creación en las artes vivas.

Para desarrollar estos puntos, esta tesis tomará tres diferentes caminos, que servirán para ampliar el campo de acción y visualización del problema comunicativo entre el artista y su interfaz; y de esta forma, caminar entre: el accionar conceptual y teórico, el técnico y de diseño, y el narrativo anecdótico más centrado en el trabajo con el usuario (artista) y su experiencia.
Esta investigación documenta un proceso colectivo, y describe el trabajo de muchas personas y de varios años; cruza por el dominio de campos tan diversos como la música, el vídeo-arte y el vídeo experimental, la ingeniería, la programación de computadores, la cultura y la

filosofía, hasta el diseño como disciplina y metodología. En el desarrollo de este trabajo el autor ha supuesto inicialmente que para un claro entendimiento del cómo afecta a la obra y al autor un cambio en la interfaz, es necesario un enfoque transversal, ya que el campo de la tecnología, y en especial la aplicación de la tecnología a las nuevas interfaces electrónicas y digitales, es tan complejo que sólo una mirada amplia es capaz de evidenciar las sutiles pero profundas implicaciones que esto conlleva en el trabajo artístico.

Este trabajo, como ya se dijo, toma varios caminos para lograr una explicación, al menos de los procesos iniciales y en general de la etapa de prototipado, en el desarrollo de interfaces para el trabajo artístico; para ello, es necesario mencionar y hacer un pequeño esbozo de uno de los puntos esenciales para el desarrollo del tema en manos del autor, el *live coding*.

El *live coding* es actualmente una gran fuente de exploración en el trabajo investigativo en el campo de las artes digitales, las artes computacionales y otros campos, tanto artísticos como filosóficos [39]. En el capítulo tres se tratará el tema con profundidad, pero para introducir sus conceptos se dará una definición inicial, el *live coding* es el arte de escribir (producir) y modificar (destruir) algoritmos en tiempo real, es decir en vivo. Este proceso no debe ser oscuro y oculto, sino más bien, transparente y en lo posible, visible a la audiencia. Esta definición es sólo una intuición del autor con el animo de contextualizar el campo de esta investigación, pero enmarcando el *live coding*, y los lenguajes de programación, dentro de las interfaces digitales.
Esta tesis no intentará explicar o identificar las cualidades intrínsecas de las interfaces digitales, pero haciendo uso de investigaciones previas como las de [39] y [10] (Esta última en el campo específico del *live coding*) buscará evidenciar cómo estas, al ser altamente maleables y con altas capacidades de personalización, muestran diferencias vitales para el trabajo contemporáneo en el arte en vivo que antes eran impensables.

Finalmente, en este texto se entenderá por artes vivas a toda manifestación artística que se realice en tiempo real, haciendo referencia tanto al proceso, como a los resultados; y para esto se explicará a continuación el concepto *live* asumido durante la realización de este trabajo.

1.1.1. Sistemas: Live

La investigación que se desarrolla en la presente tesis orbita todo el tiempo en el concepto *live* que traduciremos *"en vivo"*, esto desemboca en términos computacionales en varias miradas, una de ellas es la mirada *"on-the-fly"* [73] que permite la ejecución y edición de fragmentos de código mientras se escribe, para esto es necesario usar el concepto de *"real time response"* , de la misma forma se usarán conceptos artísticos como *performance* y *happening* para explicar y dilucidar los procesos artísticos que se desarrollan directamente frente al espectador.

Como consecuencia del trabajo colaborativo bajo el concepto *"en vivo"*, y en la perspectiva de la improvisación en el trabajo artístico, se debe tener en cuenta que la respuesta creativa está íntimamente ligada con numerosas variables entre las que puede mencionarse el público, el funcionamiento adecuado de los controles, el correcto y exacto seguimiento del guión, entre muchas otras; y dado que no hay posibilidad de post-producir el material (para eliminar posibles fallas) y al no tener esta posibilidad, el resultado, la obra misma, se convierte en un ente único e irrepetible que solo puede ser presenciado y experimentado de manera completa *en vivo*.

1.2. El campo de investigación

Esta investigación cruza por diferentes campos y toca temas diversos, pero no se puede olvidar que marco central es el arte, el artista y su obra. Aquí resalta la música, y cómo afecta la tecnología al artista ejecutor, mediante el uso de una interfaz. Pero no solo hay artistas músicos que usan interfaces en su trabajo artístico, para los artistas visuales son fundamentales para el uso de la imagen y la formación de esta en contextos *en vivo*. En este punto es claro que entramos en el campo de la interacción humano-computador (HCI, por sus siglas en inglés Human Computer Interaction), sus métodos y procesos. Entonces, al caminar por el el HCI y el *live coding* se entra en el campo de las artes computacionales y el diseño en dos vertientes, el diseño de software y el diseño industrial y objetual.

También se involucra el trabajo colaborativo y en red, que permiten un análisis metodológico para adaptar los grupos de artistas, la evolución de los dispositivos a este tipo de obras y al proceso de creación en contextos digitales grupales.

Así mismo, es importante mencionar que el contexto teórico y filosófico que es necesario para dar soporte al entramado general de la tesis y la experiencia cotidiana.

1.3. Trasfondo y Contexto

El trabajo realizado en esta tesis toma como trasfondo algunas experiencias pedagógicas. La primera es con el colectivo Algo0ritmos y la experiencia con el movimiento del Electronic Dance Music (EDM) en la ciudad de Medellín. Esta primera experiencia se abordó tanto desde el punto de vista pedagógico, como el artístico y creativo. Empezó como un taller de código en el año 2014, para evolucionar hasta convertirse primero, en un colectivo artístico con un alto componente de creación de herramientas funcionales para el *live coding*, y segundo, en un grupo musical mundialmente reconocido por su participación en festivales, conciertos y fiestas Algorave (este término se definirá ampliamente en el capítulo tres) que

hace uso de las herramientas construidas durante el proceso pedagógico.

La segunda experiencia es el trabajo con el colectivo CICLUX y el semillero de investigación-creación CINEVIVO, del Instituto Tecnológico Metropolitano de Medellín. Esta experiencia es un trabajo enfocado en lo visual, en la creación y manipulación de imagen en tiempo real, y en la construcción de interfaces personalizadas y ajustadas a cada ejecutante y a cada pieza. En esta experiencia el trabajo tiene un componente de construcción manual y desarrollo de artefactos físicos, que complementan la dimensión performática, y un componente de software que va desde la optimización de herramientas, hasta el desarrollo de nuevas formas de interacción que llevan de nuevo al *live coding*.

De igual forma, se incluye también las experiencias con el semillero de investigación-creación ACORDE del ITM y la Cybernetic Orchestra de la Universidad de McMaster, donde la composición musical en contextos contemporáneos es el centro principal de su actividad. Usando tanto técnicas de la música electroacústica como las interfaces físicas en el primer caso y el *live coding* para el segundo.

1.3.1. Trabajos que inspiraron e influenciaron

En el ámbito internacional, se debe mencionar el trabajo de Sam Aaron, su sistema de *live coding* Sonic PI [6] y sus aportes al trabajo de la pedagogía del código, siendo quizás el desarrollador del único lenguaje de programación completo, dentro del *live coding*, orientado al trabajo con niños. Thor Magnusson y su aporte en el sustento teórico para el diseño de interfaces digitales, y claro, por el proyecto Ixi[39], en especial por ixilang [40], ambos referentes mundiales tanto en la epistemología de los instrumentos digitales, como el desarrollo de lenguajes de programación para la manipulación de patrones. Alex McLean y el trabajo realizado en el campo del EDM y los lenguajes de programación, es decir, TidalCycles [49] y el movimiento Algorave; donde él es tanto el creador de las herramientas, como el principal promotor del movimiento a nivel mundial. David Ogborn y el trabajo en red, el trabajo colaborativo, los ensambles electrónicos y digitales, y su aporte para convertir el *live coding* en una disciplina de investigación [50]. David Ogborn también ha influenciado este trabajo desde las herramientas creadas por él, que permiten y facilitan los acercamientos pedagógicos, escogiendo aquellas que no necesitan de instalación para el usuario, y además permiten la interacción de grandes grupos y ensambles en la creación de piezas musicales, entre las cuales se resalta las herramientas Extramuros y Estuary [51].

Cabe resaltar también el aporte que hacen festivales, simposios y conferencias internacionales, donde se discute sobre estos temas, por eso es necesario mencionar la conferencia New Interfaces for Musical Expression - NIME (http://www.nime.org/), que se viene realizando desde 2003, y el nuevo International Conference on Live Coding - ICLC

(http://iclc.livecodenetwork.org), que para el año 2017 realiza su tercera versión.

1.3.2. El principio de la idea

Inicialmente, la necesidad de hacer herramientas personalizadas para el trabajo artístico, en este caso particular, solucionar problemas de programación orientados a la creación musical, nació como un vacío en las herramientas pedagógicas necesarias para llevar a cabo un taller en el Parque Explora de la ciudad de Medellín en el año 2014. En ese momento se convocó a un grupo de participantes, quienes además de tener diferentes edades, presentaban diferencias en sus procesos de aprendizaje y en recursos económicos, lo cual planteó el reto de guiar un curso taller de programación creativa, orientada a la música, en un contexto plural y sin ninguna herramienta que permitiera el diseño e implementación rápida de soluciones a los problemas del taller.

De esa experiencia nacieron los primeros ancestros de la familia de herramientas para el trabajo *en vivo*, y las primeras ideas para el desarrollo de la librería Chmusick.

Esos primeros experimentos fueron objetos y clases escritos para abstraer conceptos puramente musicales y llevarlos a un nivel más alto dentro de la sintaxis específica del *Domain Specific Language* (DSL) escogido para el taller: ChucK [73], una librería que empezó a ser usada de forma independiente por músicos y artistas que hicieron que evolucionara por caminos insospechados hasta alcanzar disciplinas por fuera de la música. Es desde ese momento en el que el trabajo se expandió al arte audiovisual y al desarrollo de herramientas para propósitos más generales en las artes.

1.3.3. Instrumentos, controladores, interfaces

En el contexto de esta tesis se entenderá a un instrumento como algo autónomo, capaz no sólo de permitir la comunicación entre artista y procesador, sino que él mismo tiene la capacidad de producir una salida esperada, es decir, es en si mismo, sistema de control, de procesamiento y dispositivo de salida. Es así como, el proceso creativo puede ser completado enteramente en un instrumento. Ahora, al separar de manera modular las fases del proceso completo, que son: entrada, proceso y salidas, tenemos un objeto que llamaremos controlador y que puede catalogarse como la primera fase del sistema; no tiene capacidad de procesamiento ni de salida, siendo simplemente el punto de entrada de la información suministrada. Es decir, un controlador es un artefacto que tiene como dominio la primera fase del proceso.

Para entender el objeto de estudio de este trabajo será necesario hablar de interfaces como

modelos multidimensionales, como dispositivos capas de migrar de una fase a otra de forma casi instantánea, que pueden ser vistos como instrumentos, y de esta forma centrar la atención en el proceso comunicativo hombre-maquina y las implicaciones de la capacidad de las interfaces electrónicas y digitales de ser transformadas para propósitos específicos, de forma rápida, a bajo costo y con alta precisión. Entonces, de esta forma es posible escapar de la tiranía del luthier o del diseñador externo que impone su visión del mundo, para adoptar una más cercana al interprete o usuario, y en este caso entrar en un dialogo creativo pero siendo conscientes de que no hay una visión exclusiva y personal en la creación artística, si no también, la visión del creador de la interfaz confluye en el proceso y en los resultados finales.

Diremos con Oliver que *üsar código ajeno nos provee de espacios de acción y esquemas de pensamiento, pues un lenguaje o un programa impregna o impone su propia lógica"* ... y por esto *"la tecnologías no es un ente pasivo que controlamos "*[4].

1.3.4. La violencia y el error

El aparato técnico que nos rodea está en una constante paradoja, por un lado nos facilita la adquisición de nuevas formas de hacer e interactuar, mientras nos limita en sus posibilidades y en el *como hacer* estas nuevas formas adquiridas. De esta manera, es el dispositivo mismo el que nos dicta nuevas formas de imaginar y nos limita con su imposición inherente a su esencia[30]. Es por esto que esta paradoja nos enfrenta ante la posibilidad de la evolución misma del mundo técnico, como una linea dibujada por los responsables del diseño, donde la materialización de los avances se limita ante la perspectiva pasiva de lo que el sistema de mercado nos pueda ofrecer, cuando ese sistema técnico es claramente activo, por no decir político.

Ante esto, aparece el concepto de violencia visto desde varias perspectivas, como un facilitador de cambios (bien sean formales, o de fondo) y el error y el *glitch* (error no fundamental o que no interrumpen el sistema principal, pero si es notorio) como puerta hacia mundos no concebidos. De esto se hablará en el capítulo Marco pero no de forma concreta sino más bien, con el animo de ocultar pistas a lo largo del texto y permitir que sea el lector el que se forme una idea de como se concibe este concepto para el análisis y evolución de el componente teórico, en el trabajo con las interfaces y la obra en si misma.

1.4. La pregunta

Dadas las características interdisciplinarias de este trabajo, es necesario plantear preguntas de investigación usando varias disciplinas y diferentes perspectivas, por lo que es inevitable que *la pregunta* asuma varias personalidades sin dejar de ser una sola hipótesis, que se muestra a si misma como intuiciones diversas del autor, según el enfoque desde el que se trabajará.

Así mismo se emprende el análisis desde la perspectiva que da la investigación-creación dado que el proceso teórico, la creación de obra y el desarrollo de herramientas se tejen como hilos de una misma tela, entendiendo su indisolubilidad y aprovechando esto mismo como licencia para introducir un análisis y formalización completo, no en el sentido científico del método tradicional, sino más bien como un antimétodo [29] que se aprovechará de visiones accidentales, errores y *glitches*, para revisarse a si mismo como obra y como producto de nuevo conocimiento.
Es la obra misma en la ejecución práctica y en vivo la que alimenta la observación y como creación artística, se deja una libertad explicita al proceso creativo que en este caso es el mismo proceso de investigación.
La heterogeneidad de los grupos, las herramientas y los resultados son entonces parte constitutiva de la obra, y el proceso de formalización escrita es una (sino la mejor) metodología valida y aceptada en la investigación en artes y no por su subjetividad deja de ser nuevo conocimiento valido.

Es entonces importante desdoblar la pregunta/hipótesis en varios enfoques:
Desde el enfoque conceptual,

- ¿Existe una relación entre la evolución y los cambios que tienen las interfaces usadas y la obra producida o la concepción del artista sobre la obra?

Desde el enfoque técnico,

- Dado que los dispositivos digitales tienen una gran capacidad de ser transformados cuando se piensa en *open source* ¿Cómo cambia la obra y la concepción del artista sobre esta de la mano de la depuración y las necesidades de programación, hardware y los cambios realizados durante el proceso creativo?

Desde el live coding visto como interfaz,

- Cada lenguaje propone una forma de ser ejecutado, pero al mismo tiempo una capacidad de ser expandido y personalizado. ¿Este proceso de personalización y expansión afecta el estilo mismo de la producción? o mejor, cada nueva característica, bien sea estética, sintáctica o semántica ¿qué cambios produce en el estilo del autor y la obra misma?

Desde la experiencia de trabajo colectivo y el proceso colaborativo,

- ¿Cómo puede el cambio de una herramienta de producción artística afectar la dinámica de trabajo en la realización de un montaje/obra?

1.5. Miradas sobre el problema

Cada capítulo de esta tesis se presenta como una perspectiva diferente al mismo problema, y de esta forma intenta explicar los fenómenos observados usando diferentes disciplinas y enfoques. La mirada está divida en cuatro posiciones que, de modo general, pueden explicarse de la siguiente forma:

- La mirada conceptual y el diseño como herramienta, para ver el fenómeno desde un punto de vista teórico, teniendo en cuenta que la exploración de esta tesis es en esencia práctica.
- El *live coding* como concepto permite una forma única de enfrentar el problema, ya que por definición misma, se plantea el trabajo en vivo, la transparencia y la evidencia del error, el trabajo en red y el diseño de interfaces (en este caso lenguajes de programación vistos en principio como interfaces musicales) como pilares en la expresión artística.
- El proceso estrictamente técnico en el desarrollo de herramientas computacionales, para resolver problemas artísticos y creativos puede verse como una ejemplificación y al mismo tiempo puede servir de modelo mental para entender el resultado final.
- El trabajo creativo en colaboración y el arte en red, que pueden aportar la mirada experiencial y mostrar una metodología para incorporar la evolución de la interfaz al proceso creativo.

Estas son, a grandes rasgos, las miradas que se aplicaron al trabajo de investigación que se describe en el presente documento.

1.6. Los objetivos

Luego de mostrar las cuatro miradas que se usarán a lo largo de esta tesis para enfrentar las perspectivas planteadas en la pregunta de investigación, y de plantear las miradas que abordarán el camino de investigación, es necesario exhibir los objetivos (específicos), que en este texto servirán como un paso a paso en la argumentación; como pistas que ayudarán a romper el objetivo (general) en secciones de análisis más pequeñas, y que además servirán para entretejer la argumentación teórica entre la urdimbre de la practica artística y la creación de interfaces, en el día a día creativo de los diferentes colectivos artísticos conectados a esta investigación, y el trabajo del autor de esta tesis como facilitador y artesano a la hora de hacer concretas las ideas colectivas. Dicho esto, el objetivo general de este trabajo es:

> Analizar cómo los cambios en las interfaces producen resultados notorios en la obra, especialmente en el arte en vivo.

Claramente se observan variables de investigación amplias, las interfaces, variable independiente, que afectan la obra de arte, variable dependiente; lo cual brinda la oportunidad para el discurrir argumentativo, ya que se podrá saltar de lo teórico a lo práctico sin previo aviso, permitiendo al autor explicar la evolución de la interfaz, la obra, el artista y el artesano desde puntos de vista diversos, casi contrarios, sin encontrar en esto dificultad, o al menos, sin tener que buscar demasiado.

Ahora, dicho esto, se puede decir que una de las ideas e intuiciones que el autor lanzará como uno de esos caminos para la explicación y análisis de este objetivo es el primer objetivo especifico.

> Explicar cómo la violencia que induce la interfaz misma afecta y permea las realizaciones que se hagan con ella.

En el uso de este concepto - la violencia - habrá cierta libertad, y se aplicarán según sea el caso definiciones fuertes y débiles de este concepto.

No es una idea original, como puede verse en el análisis lingüístico [3], en los estudios de comunicación y nuevos medios [66] y en campos como la física, donde el cambio de paradigma hacia la cuántica, con nuevos usos en las palabras, técnicas y enfoques plantea nuevos retos, y claramente afecta los cimientos de nuestro conocimiento de (podría afirmarse) la realidad misma[68].

Para enmarcar el cuadro que se nos presenta, esta tesis se limitará al concepto de la programación al vuelo o *live coding* dando primero un bosquejo general y luego usando los tonos y colores derivados de esta exposición, lo que permite plantear el segundo objetivo específico.

> Resaltar aquellos conceptos que nacieron como revolución contra la interfaz, en los lenguajes diseñados para el *live coding*.

La revolución se entenderá como un detenerse violentamente y transformarse o escaparse, y por esta razón se remarcarán estos conceptos para usarlos como puertas de salida sugiriendo de esta forma una metodología del error (idea ya sugerida en [29]), que posiblemente desemboque en futuros trabajos en una política del error y el glitch.

Así mismo, es necesario entender como estos conceptos se relacionan y pueden servir como pilar argumentativo a la hora de llevar estas ideas a la obra y a la creación artística, lo que plantea el tercer objetivo específico.

Comparar la experiencia en el uso de las interfaces, tanto virtuales (GUI y textuales) como físicas de manera experimental.

Es decir, en la obra en vivo, aquella que se hace frente al público, que se pone de pie desnuda frente a la audiencia. Y es que no hay mejor experimento en el arte que aquel donde el artista/artesano muestra todas sus cartas sometido al caos de miles de variables en escena.

1.7. La contribución

Este trabajo ha sido, inicialmente, producto de un esfuerzo pedagógico y artístico, y luego de un largo proceso, un trabajo con aportes técnicos desde lo computacional, la electrónica y el diseño. Es necesario mencionar que los resultados de este trabajo no sólo están redactados en esta tesis, sino que son y han sido la herramienta de trabajo de una gran cantidad de personas y colectivos, no sólo a nivel local. También es posible rastrear colectivos y artistas de otras latitudes que están en constante contacto y hacen uso de esas herramientas, siendo de gran ayuda en el proceso de depuración y prueba.

Es por esto que las contribuciones de esta tesis pueden mencionarse en dos ámbitos diferentes, el teórico y el práctico.

- Un contexto y un marco de trabajo (Capítulo 2) en el cual se explicarán los fenómenos y las interacciones entre hombre-maquina desde el concepto "techné" como un englobador, como una vuelta al principio, a la concepción griega donde el arte tenia un componente de construcción de sus propias herramientas, del trabajo manual (que en este caso puede ser simbólico pero al mismo tiempo requerir del esfuerzo de la artesanía). Así mismo, este concepto servirá como guía de trabajo.
- Un estado del arte sobre el *live coding*, disciplina que por su actualidad y relevancia en el campo de las artes digitales y por los puentes que tiende entre estos conceptos descritos en el punto anterior necesitaba de este compendio (Capítulo 3).
- En el campo de las artes computacionales aporta con la exploración hecha en formas alternativas de interacción y de sintaxis para DSLs, trabajo en respuesta en tiempo real de sistemas computacionales y sistemas alternativos en el diseño e implementación de lenguajes de programación y sus componentes.
  Además presenta un marco que puede servir como ejemplo para trabajos futuros en el campo de la programación creativa.
- Un contexto de trabajo en el cual explicar y entender el trabajo colaborativo y como el trabajo en red aporta metodologías de trabajo y montaje que ya han sido probadas y pueden servir como modelo a otros colectivos artísticos.

1.7.1. Interfaces generadas

En esta sección se mostrarán los aspectos claves que servirán como guía e introducción para el resto de este documento en lo que compete a las interfaces resultado de la investigación, actualmente, algunas de las aplicaciones tienen varias versiones, todas en uso, pero mantienen una idea general en cuanto a sus conceptos relevantes.

CQenze

En palabras generales CQenze es un DSL, que es a la vez dos cosas diferenciadas conceptualmente: un lenguaje simple de programación y un secuenciador de pasos para muestras de audio. Como lenguaje carece de estructuras convencionales como bloques o estructuras de control, esto sucede a causa del segundo concepto; al ser un secuenciador no requiere estructuras de control temporal, ya que estas están incorporadas en el lenguaje mismo, como características sintácticas. Este DSL nació como lenguaje de primera experiencia, es decir, no programadores que se enfrentan por primera vez a la experiencia de escribir código.

Chmusick

Chmusick es actualmente, una librería dividida en dos modelos sintácticos diferentes el original está escrito usando el modelo mismo del lenguaje que pretende extender (chucK), es decir Pogramación Orientada a Objetos (OOP, del ingles Object Oriented Programming), usando una sintaxis similar de lenguajes clásicos como Java o C#. El segundo modelo se acerca a la sintaxis de lenguajes funcionales como Tidal. Cabe resaltar que aunque su modelo de construcción es funcional, su sintaxis esta construida como objetos y clases.

CineVivo

CineVivo es un software escrito en C/C++, el cual usa toda la potencia, flexibilidad y estabilidad de openFrameworks[70]. En su última versión, CineVivo tiene un pequeño editor de texto que permite que las funciones del Software sean controlas usando código en tiempo real, lo que lo convierte en un DSL para el live coding de videos.

CineVivo MAP

CineVivo MAP es un software escrito sobre C/C++, el cual incluye una Interfaz Gráfica de Usuario (GUI, de sus siglas en inglés), aunque también puede ser operado usando controladores MIDI (Musical Interface for Digital Instruments), microcontroladores del tipo Arduino o

Raspberry PI, o usando protocolos Open Sound Control (OSC) y User Data Protocol (UDP). Lo importante en este software es su capacidad para ser controlado casi de cualquier forma, y con cualquier tipo de dispositivo, permitiendo al artista ejecutor ser creativo a la hora de hacer su *performance*.

Interfaces Físicas

En el proceso de desarrollo de las herramientas para usuarios (artistas) no involucrados directamente con el *live coding*, se desarrollaron también interfaces físicas de control para facilitar ciertos aspectos de la ejecución en vivo. Estas hacen parte del trabajo especifico con CineVivo y CineVivo MAP.

1.8. A donde ha viajado este trabajo

Como parte de la socialización y divulgación de esta investigación y sus contribuciones, las herramientas, interfaces desarrolladas y resultados de este proyecto han sido parte de la maleta de viaje de muchos artistas, así mismo el autor de este documento ha viajado por sus propios medios a diferentes presentaciones artísticas y eventos académicos, por esta razón algunas secciones de esta tesis son tomadas de artículos ya publicados, en proceso de publicación y han pasado por revisión de pares académicos.

En particular, el trabajo sobre CQenze es usado como punto de partida para todo el capítulo dos, desde la perspectiva de diseño y arquitectura de software, así mismo sus aportes pedagógicos son puestos en el capítulo uno. Estas ideas fueron presentadas en:

- Betancur, Esteban. 2016. *"Diseño e implementación de un DSL : CQenze, como lenguaje de primera experiencia para el código en vivo"*. Presented at the 15th Festival Internacional de la Imagen, Manizales, Colombia. May 9–13.
- Ogborn, D. Beverley, J. Navarro, L. Tsabary,E. McLean, A. Betancur E. 2017. *"Estuary: Browser-based Collaborative Projectional Live Coding of Musical Patterns"*. Presented at the ICLC 2017, Morelia, México, Dec 4-8.

La idea de CineVivo, todo su trasfondo de diseño y el trasegar de este software de prototipo a artefacto funcional y herramienta artística fueron expuestos en:

- Betancur, Esteban. Florez, Erica. 2017. *"Customized Cinema, Proceso de diseño e implementación del software CineVivo para el desarrollo de piezas visuales en tiempo real"*. Presented at the ISEA 2017, Manizales, Colombia. June 11-18.

Una explicación más orientada a la relación sintáctico-semántica de Chmusick en:

- Betancur, Esteban. 2017. *"Chmusick, una librería para convertir chucK en un lenguaje tipo Algorave"*. Presented at the ICLC 2017, Morelia, México, Dec 4-8..

Adicionalmente, las herramientas implementadas en el desarrollo de esta tesis, han sido parte de numerosas presentaciones y *performances*, entre los que cabe mencionar:

- Museo de Antioquia, Medellín, Patio Sonoro 2016.
- Festival Internacional de la Imagen, Manizales 2016.
- MAMM 2016, clausura encuentros sonoros.
- ICLC '16, Canadá, Obra "Bizcocho de Teja".
- Festival de Cine Corto, Popayán, Obra "Nihil Cinema " 2016.
- Algorave, Wearefive, 2017 Streaming global.
- Algorave, Algorumba, 2017, primer encuentro Algorave de Latinoamérica.
- ISEA '17, Manizalez, Obra "Leviathan" con el colectivo mexicano Andamio.
- ISEA '17, Manizales, Obra "Ruido" colectivo CICLUX.
- ICLC '17, Morelia, México, Obra "Random Corpus Binary" Semillero Cinevivo.
- ICLC '17, Morelia, México, Obra "Algoritmos" colectivo artístico Algo0ritmos.
- Experiencias de diseño UPB, Medellín, Colombia.
- Reudo'18, Medellín, Colombia.
- NIME '18, Virginia Tech, Virginia, USA
- DanceCult, Artículo a ser publicado en la edición de Noviembre de 2018.

Y otros.

Como anexo a esta tesis se presentan algunas fotografías y sitios web donde están las procesos de montaje, prototipos, repositorios de software y demás elementos que se consideraron importantes.

1.9. Guía de contenido

El capítulo dos presenta una revisión general de conceptos, para hacer precisiones que se consideran importantes. Es un capítulo se plantea la discusión tecnológica desde un punto de vista conceptual, se hablará de los procesos pedagógicos, de cómo la interfaz digital puede plantear soluciones a cuestiones ampliamente discutidas en el campo del diseño y de cómo, de forma audaz, también puede enfrentar al usuario ante sí mismo.

Se discutirá cómo la digitalización transforma al usuario con un poco de práctica en diseñador y luthier, y esto hace que rápidamente la imposición violenta de otras visiones del mundo, diferentes a la suya, puedan ser enfrentadas usando el error y la experimentación, haciendo que este nuevo constructor busque la personalización, el *Do it Yourself* (DIY) y otras tendencias de este tipo para encontrar su propia voz.

El tercer capítulo es un recorrido por el *live coding*, entendido como interfaz. Este capítulo se puede entender como un estado del arte, pero al mismo tiempo, es una revisión de los horizontes del concepto y su ampliación al uso del *live coding* como interfaz, para el trabajo en otros campos diferentes a los que han sido su cuna (la música, en especial el Electronic Dance Music - EDM). Este capítulo responde a la contextualización del *live coding*, que permitirá mejorar la comprensión general del por qué se introducen cambios en el proceso de evolución de las interfaces.

El capítulo cuatro es un capítulo técnico-anecdótico, desde la perspectiva del diseño y la arquitectura de software, junto a la experiencia en el trabajo con los artistas. Es necesario hacer hincapié que, aunque se tocan los conceptos computacionales y de diseño, el autor intenta mantener un lenguaje accesible a los lectores no expertos, sacrificando la profundidad en la arquitectura. Esta decisión se tomó ya que todo el código es de acceso público y se incluye una referencia a la ubicación en la red como un anexo en la presente tesis, así que el lector interesado podrá explorar el código a libertad.

Para este trabajo se plantearon tres ejercicios prácticos, CQenze, un DSL diseñado con propósitos claramente pedagógicos, Chmusick, una librería que expande las posibilidades de el lenguaje de domino especifico chucK y CineVivo como herramienta para el desarrollo de piezas de live cinema. Estos servirán como entidades ejemplificadoras de los conceptos, además ya fueron usados en campo, en talleres y obras artísticas presentadas en eventos nacionales e internacionales, evidencia que le confiere un especial interés al recuento de estos casos.

El capítulo cinco es un capítulo que cumple con el objetivo de evaluar y comparar el proceso de introducción de cambios en la interfaz, como un proceso donde el artista, la interfaz y el

desarrollador, se involucran personalmente en la construcción de las herramientas que mejor se adapten al proceso creativo, introduciendo pequeños cambios luego de probar en vivo, y descubriendo puntos donde la comunicación entre los involucrados podía ser mejorada, reduciendo la necesidad de adaptarse, y más bien adaptando la interfaz paso a paso en un proceso de continuo mejoramiento y aprendizaje. Por último, en el capítulo seis se presentan las conclusiones y reflexiones del trabajo desarrollado.

2. Marco e ideas

2.1. Resumen

Este capítulo presenta el marco, el contexto y algunas ideas necesarias para entender el desarrollo detrás de las interfaces construidas, para los procesos de varios colectivos; ofreciendo una panorámica general a la visión artística, el uso de los nuevos medios y las tecnologías desarrolladas o adaptadas en el transcurso de esta investigación. Es así como se explicarán algunos conceptos claves que servirán como guías para el resto de la lectura, lo que permitirá entender y visualizar el trabajo realizado, el error, la violencia y tiranía de la interfaz y su la personalización, llegando hasta la adaptación y mutabilidad que está inmersa de forma natural en la misma naturaleza de los nuevos medios.

No se busca establecer un contexto ni un marco para comprender la generalidad de los proceso artísticos y creativos, sino más bien uno particular a los desarrollados durante la presente investigación; sin categorizarlos por los medios usados, mostrando los procesos y las transformaciones que fueron resultado de evoluciones particulares en contextos únicos.

2.2. Marco de Referencia

Desde el momento en el que una especie de mamífero descubre que su limitada capacidad física y mental puede ser conectada a través de interfaces con su exterior, y que esta posibilidad puede expandir su habilidad al punto de sobrepasar sus límites con cada nuevo descubrimiento, el homo-sapiens (nombre de esta especie) se transforma en un generador de conexiones, y por lo tanto de nuevas perspectivas, que se han ido transformando con el paso del tiempo; trastocando no solo el contexto que habita, sino también su cuerpo y mente, y lo más importante, sacando del eje su punto de vista, haciendo de ese cambio en la perspectiva su más grande potencial transformador y el punto de apoyo donde se ha anclado la palanca de su adaptación y evolución. La capacidad de hacer abstracciones a modo de palabras y del lenguaje en si mismo, es quizás una de las más potentes herramientas que el humano tiene a su disposición para sobrepasar los límites corporales y crear nuevos mundos que se construyen y destruyen en instantes tan cortos como las palabras que los contienen.

Pero no es necesario viajar tan lejos, es posible enfocar el marco de trabajo en un alcance

más concreto y es por esto que el trabajo se puede situar en el marco digital y electrónico, y con esto en una herramienta técnica especifica. Por esto, se puede empezar diciendo que para los propósitos de esta investigación *"la informática o computación es la ciencia del tratamiento automático de la información mediante un ordenador"*[54].

El computador como herramienta técnica desde sus inicios muestra su potencial para la producción de imágenes y sonido, por su capacidad para hacer cálculos a gran velocidad, y no sólo esto, sino su habilidad para producir y generar tanto formas de procesar lo ya existente, como de generar nuevas formas de producción artística. Ejemplos de esto pueden verse en el arte generativo, donde artistas como Brian Eno, de la mano de sistemas computacionales y algoritmos generativos pueden realizar obras con características únicas, no solo como novedad inicial, sino en cada nueva ejecución de los algoritmos, produciendo de esta forma infinitas combinaciones musicales [2]. Así, el arte *"acontecía calculable y, por lo tanto, incorporaba nuevas propiedades estéticas inéditas"* [54].

2.2.1. Techné

Para entender el marco de esta investigación se requiere explicar de forma breve el enfoque altamente práctico que se escogió; y es que más que introducir unos conceptos para ir desarrollando una argumentación, una epistemología de los instrumentos digitales [39], esta investigación propone un enfoque desde el hacer, desde la prueba y el error, desde lo artesanal, y por esto se establecerá como guía conceptual el techné griego, re-evaluando también de manera indirecta, pero consciente, la supuesta superioridad del pensar sobre el hacer, expuesta en la mirada occidental traída a nosotros desde la filosofía de Aristóteles.

2.2.2. Dédalo -el hacedor-

...y el alma humana reposando en la memoria de lo -hecho-

Dédalo era el constructor griego por excelencia, en su taller materializaba desde el caos informe objetos míticos que tomaban forma en sus manos de artesano-artista-constructo, para habitar la realidad circundante. Entre sus creaciones se pueden mencionar el laberinto para encerrar al minotauro, y las famosas alas que elevaron a su hijo Icaro hasta el sol, pero entre todas las historias hay una especialmente importante para entender lo que significa, y lo que conlleva esta idea de dar vida a nuevos seres hijos de la técnica usando las manos, y esta es la historia de Pasífae.

Dédalo construye una vaca hueca, donde Pasífae puede ocultarse, logrando de esta forma saciar su deseo sexual con uno toro, experiencia de la cual nace el Minotauro. Aquí surgen

dos ideas fundamentales, la primeroa, donde el artesano-artista-constructor es quien tiene la habilidad de materializar los monstruos (seres inexistentes, ideas complejas) con la ayuda del trabajo de sus manos, uniendo (construyendo la interfaz) realidades paralelas y por lo tanto desconectadas con su ingenio y habilidad manual, estableciendo una comunicación entre mundos que de otra forma hubieran permanecido desconectados.

Junto a esta primera idea es importante conectar otra. Como ya se dijo, el hacedor esta encargado de construir la interfaz, pero luego, esa interfaz tiene una existencia propia, una que además, está cargada con la memoria tanto de su creador como de sus antepasados [28], la porta en su esencia, que es su forma material.

Acá el concepto de techné puede entenderse entonces no sólo como una habilidad netamente práctica para la fabricación de artefactos, sino como una forma de conocimiento altamente práctica, una alquimia entre arte y ciencia, donde el artesano está consciente de su trabajo como enlazador de mundos, tendiendo puentes entre las ideas y el mundo material. El artista entonces, se convierte en el encargado de la poiesis, es quien debe, *"traer a la existencia aquello que no existía"* [55], emergiendo como Dédalo se sus obras, sus creaciones, sus monstruos.

2.2.3. Lenguajes de Programación

Los lenguajes de programación son la interfaz más importante para establecer comunicación entre quien desea desarrollar una tarea computacional y la máquina como tal, es decir, un lenguaje de programación funciona como mediador entre las intenciones humanas y las instrucciones de bits que son las únicas que tienen sentido para un computador. *"Son el más general pero al mismo tiempo la más intima y precisa herramienta para instruir un computador"* [72].

Además de esta definición, es importante saber que con la llegada de computadores más potentes, capaces de desarrollar tareas en tiempo real, fue posible equiparar conceptos de otras áreas a las ciencias computacionales; conceptos como la interpretación, la expresividad discursiva entendida desde el hecho en vivo, y otros más. Pero aún suena extraño decir *"soy un interprete de computador"* ó *"toco computador en un ensamble"*, si se quiere entender el computador como una interfaz, y más aun, si la interfaz no es ni siquiera el computador como hardware sino el lenguaje de programación interpretado en vivo. Es aquí donde los lenguajes de programación se muestran como un puente y además sirven para evidenciar cómo la percepción misma debe adaptarse al lenguaje en el que va a ser expresada la imagen (no solo visual sino en concepto amplio) percibida; *"si un lenguaje de programación no cambia tu forma de ver el mundo no vale la pena aprenderlo"* [72].

Por esta razón, al hablar de cambio y transformación, se vuelve necesario hablar de violencia vista como factor inductor de cambio. Así, el problema de la interfaz debe ser entendido como un problema holístico[72] y más cuando el sistema en cuestión no tiene tiempo de corregir sus errores (preproducción, producción, posproducción) en amplios espacios de tiempo, sino que debe adaptarse rápidamente a las necesidades del entorno (live acts - artes vivas); es decir, la evolución/adaptación del sistema está por encima de los costos resultantes de estos cambios.

Lo anterior permite concluir que, este tipo de adaptación rápida debe ser explicada desde la experimentación misma, dado que es una característica intrínseca de los sistemas caóticos, y es justamente su inestabilidad y constante cambio lo que hace que sean importantes para el desarrollo; es decir, la interfaz para el artista digital *en vivo*, no debe ser rígida y estática, bien sea una interfaz textual (lenguajes de programación), una interfaz gráfica de usuario (GUI) ó una interfaz física (controlador).

Para terminar esta sección, es posible afirmar, como lo hace Magnusson [42], que el usuario se transforma en desarrollador, y con el uso de software como Pure Data, SuperCollider, CSound, Max, ChucK, JavaScript, y hardware como Arduino y Raspberry Pi, un mundo se ha abierto para la creación de nuevas tecnologías.

2.2.4. Microcontroladores

Un microcontrolador es un chip con capacidad de computo, memoria y una serie de entradas y salidas. Aunque este tipo de dispositivos han estado en el mercado por mucho tiempo, no fue sino hasta el desarrollo y aparición del Arduino (del cual se hablará más adelante) que el uso de microcontroladores se hizo de uso masivo en las artes, dado que estaba construido justamente para simplificar el uso sin necesidad de pasar mucho tiempo en los detalles del hardware. Por esta razón el uso de este tipo de microcontroladores no requieren de un público especializado en electrónica, programación o diseño, siendo además una alternativa económica que da la posibilidad de usar y modificar hardware computacional fácil y económicamente [32].

Además de su economía y las diferentes ventajas técnicas que ofrece, los microcontroladores Arduino están inmersos en la cultura de desarrollo libre, lo que significa que la evolución de la plataforma misma está enmarcada en la prueba y el error como principio fundamental, por lo que una comunidad bastante activa es la que se encarga de producir tanto la documentación del sistema, como nuevas librerías y prestaciones. Esta comunidad está conformada por personas de disciplinas diversas; desde ingenieros, artistas hasta diseñadores. Equipos multidiciplinarios trabajando alrededor de Arduino en ambientes de aprendizaje y otros contextos. [32].

Existen otras opciones de hardware que cumplen funciones similares, con pequeños cambios incorporando otras funcionalidades. Se mencionará también el computador tipo placa simple de bajo costo llamado Raspberry PI (RPI), dado que este pequeño computador, fundamentalmente en la misma filosofía que el Arduino, permite el acceso a herramientas de programación avanzadas con una inversión mínima, además, esta plataforma de desarrollo sirve como huésped de Sonic PI [6], lenguaje para el live coding del que se hablará más adelante.

2.3. Ideas

2.3.1. El error como elemento expresivo

En el trabajo artístico y creativo se creía que las ideas, en un estilo platónico, habitaban mundos perfectos, donde cada cierto tiempo permitían la entrada de algún elegido para que accediera al conocimiento y de esta forma, por gracia de alguna musa, el artista lleno de inspiración ejecutara su obra magna. Esta concepción, que además ha sido perjudicial para el arte mismo, alejándolo de la humanidad tan propia, dejó de lado todo el componente técnico, haciendo ver las artes como cosas de dioses que emergían de la nada, cuando realmente el proceso artístico requiere de tiempo, trabajo y siempre está enmarcado en la mutabilidad, y la evolución, tanto de las ideas como del método de ejecución y materialización.

Es en este punto donde puede verse la posibilidad del error como factor primordial para el aprendizaje [56] ya que éste facilitará desligar el arte del mundo de los dioses para traerlo de vuelta a la humanidad. Además, si se está dispuesto a extender la metáfora de forma agresiva, el error puede llegar a conformarse como elemento mismo de la expresividad natural, tanto del arte como de la ciencia en el proceso de construcción de conocimiento [44].

No se trata de usar el error como cliché, diciendo simplemente que *"de los errores se aprende"*, es tomar el error mismo y usarlo como herramienta evolutiva dentro del discurso. El arte en este caso, a diferencia de otros campos del conocimiento humano, no debe preocupase de interpretar o representar (menos aún, de encontrar) la verdad tras una teoría o explicación de la realidad. Por lo que decir que el camino del error trae consigo un alto riesgo de equivocarse [44], en este caso es trivial y abre la puerta a una exploración compleja donde el conocimiento, la habilidad y el error (ahora como parte del discurso, plenamente aceptado tanto por el artista, como por su interfaz y su audiencia), descubren nuevas formas de enfocar su flujo creativo sin tener que fingir inspiración divina, humanizando de nuevo el arte (sin mencionar que los resultados derivados de una exploración abierta al error pueden alcanzar lugares inesperados).

2.3.2. La violencia y la tiranía de la interfaz

La interfaz es el puente entre dos mundos, ambos cargados de la memoria de los siglos. Uno, el ser humano, que construye su identidad personal y por lo tanto su necesidad expresiva con el código biológico que trae impreso en su ADN, sumándole a esto su posible interpretación de la realidad social, cultural y tangible que se hace idea usando una primera interfaz que es el lenguaje, y las limitaciones que este pueda tener, es decir, nuestra concepción de la realidad está mediada por una interfaz, el lenguaje, que esta limitado por su misma estructura, su incompletitud y la introducción del ruido que conlleva la experiencia personal en relación a la formación de ideas (palabras) en el lenguaje que aprendemos y que se establece como la única forma de entender en mundo. El segundo mundo es el máximo ejemplo ("metáfora de metáforas" [20]) del proceso de virtualización de la memoria humana [28], el computador y todo lo que el sistema digital involucra, código-software-lenguaje, red-conexión, hardware-prótesis, etc. [15]

Ahora, este sistema digital, es interfaz de sí mismo, y a su vez, cada parte del sistema puede ser entendida como una nueva interfaz para otra parte del mismo, lo que hace de este un sistema altamente complejo [15]. Esto puede estudiarse en la literatura citada, pero en esta investigación vale la pena resaltar un punto que está asociado además con el sistema de producción, donde los artefactos se producen de forma masiva, de forma genérica. Toda interfaz es violenta por sí misma, es decir, impone su naturaleza como una distancia entre el artista y su obra. Una forma de ver esto es ver la interfaz como un lente de color, la realidad en sí misma (dando por sentado que existe tal cosa) ha de ser interpretada por un sujeto, pero este sujeto sólo puede mirar a través de este lente que es la interfaz, lo que transformará su idea de la realidad tiñéndola con el matiz que la interfaz le impone. De cualquier forma, las ideas que el sujeto se forme de tal realidad, siempre estarán mediadas por la interfaz. Si además esta interfaz es un diseño genérico, que pretende responder a una necesidad expresiva (de adentro hacia afuera) o interpretativa (de afuera hacia adentro) su naturaleza violenta tiende a hacerse más notoria, dado que en la relación con el usuario, es este quien debe adaptarse a las condiciones impuestas por la interfaz; y por lo tanto, también por las condiciones e intereses de los involucrados en el diseño, desarrollo y producción de la misma.

Es así como el sistema productivo impone una forma de hacer las cosas, y para concluir, aventurarse a preguntar también, si los lenguajes de programación, inmersos en las mismas dinámicas, al ser estos mismos una interfaz, no imponen también una visión hegemónica del mundo (lenguajes militaristas - imperativos) donde se tiende a excluir concepciones tanto minoritarias, como aquellas tradicionalmente alejadas, las no occidentales o las vistas como débiles[20].

Historia I - El código dador de vida

Rabí - dijo el pequeño a los pies del cúmulo de barro ante el que trabajaba el maestro - ¿puede decirme que es lo que escribe usted en la frente de este monstruo? Rabí Abraham, sin poner mucha atención en sus herramientas, mira al pequeño y con un gesto bastante histriónico, traza en el aire, en una ceremonia silenciosa, tres grandes movimientos, cada uno perfectamente diferenciado del otro... al terminar, mirando al cielo, pronuncia el salmo del profeta - que Jacob ya sabia a sus ocho años - y de nuevo mira al niño, con un pequeño soplo de aire sale de su boca la palabra "Verdad". El golem, el protector del pueblo era, tan solo barro así como Adam, los dos eran polvo de la tierra que inerte, se movía por la voluntad de las fuerzas naturales, pero el destino de la tierra misma fue cambiado dándoles un soplo de vida, un poco del algoritmo universal del movimiento y el caos. Las letras hebreas, contenedoras en sí mismas, de la potencia del movimiento y la vida en un caso sopladas sobre Adam, en el otro, escritas sobre la frente del golem, actúan como la llave que de forma trascendente dan al barro la potencia de generar movimiento.

3. Live Coding: Un recorrido por el concepto

Este capítulo presenta un estado del arte del concepto *"live coding"* también llamado *"on-the-fly programming"*. La revisión va desde sus inicios como arte computacional, sin dejar de lado precedentes más antiguos, pero centrándose en el periodo que nace con el trabajo de McLean, Collins, Rohruber y Ward (2003) y Cook y Wang (2004), hasta el presente. La revisión se plantea desde varios frentes como son: el trabajo filosófico, conceptual y los esfuerzos de formalización académica, el *live coding* como arte computacional, las principales plataformas junto a sus principales desarrollos en las artes visuales, música, pedagogía y artes computacionales, sin olvidar todas las iniciativas no digitales que han servido de campo de exploración y por último el trabajo en red, las plataformas colaborativas, los ensambles y orquestas que basan su trabajo en el uso de computadores como instrumentos musicales o como plataforma principal de expresión artística.

Adicionalmente, se muestra el gran potencial de este medio técnico como lenguaje expresivo y el cómo ha permeado y ha sido permeado por diferentes enfoques, disciplinas artísticas y desarrollos computacionales. Así mismo, se resumen los procesos que han ido llevando al concepto por un proceso de naturalización, y adaptación como herramienta de expresión tanto para las artes como para la educación y el desarrollo de nuevos modelos para la implementación de lenguajes de programación de dominio específico.

3.1. Preludio

El *live coding* ha sido definido de muchas formas y a lo largo de esta revisión se explorarán algunas de estas definiciones, para que sea a través de este esfuerzo de formalización que se permita establecer un camino para entender la práctica que hasta el día de hoy se ha denominado *live coding*. De todas formas, es necesario empezar con una idea, un concepto que sirva de germen, y para eso diremos con Collins que *"en este texto supuestamente lineal, se explorará la compulsión artística a cambiar de parecer"* [21] y aprovechar estas palabras para tomar ciertas libertades literarias en la forma de escribir esta revisión y reclamar el derecho a cambiar de idea o de estilo en medio del discurso, al mejor estilo del *live coding*.

De esta primera intuición sobre lo que es el *live coding* es importante notar que este puede verse como el derecho de quien se expresa a cambiar de opinión, y más que un derecho, es la compulsión interna a hacerlo constantemente. Así mismo, y contrario a lo que se espera de este texto, es posible que el lector desprevenido sienta que el autor da saltos de un tema a otro evitando la linealidad, y tendrá razón, dado que este texto está concebido usando la tradición formal del *live coding*, que le permite cambiar de opinión o saltar en el proceso durante la escritura. De igual forma, la definición completa (si es que esta puede darse sobre el *live coding*) sólo aparecerá hasta el final, de forma parcial y nunca definitiva, ya que el presente capítulo se construirá desde tablero en blanco a modo de homenaje al concepto mismo.

Sin entrar en detalles por el momento, el *live coding* o la programación en tiempo real puede entenderse como un movimiento [41] o práctica-acción artística contemporánea [21] donde los algoritmos juegan un rol

principal junto a quien propone estos algoritmos, los modifica, ejecuta o (y esto será discutido posteriormente) "simplemente" los lee en tiempo real[41][67].

Con esta introducción en mente, el resto de este texto se organizará de la siguiente forma: en la siguiente sección se revisan los antecedentes históricos y los precursores, así como revisiones previas, formalizaciones y trabajos conceptuales y filosóficos. Luego, se mostrarán las tendencias y usos del concepto, en cuatro frentes que son: 1. Los acercamientos pedagógicos 2. Referentes artísticos bien sea computacionales o no, en esta sección se habla sobre el movimiento Algorave. 3. Una revisión general de los principales sistemas para el *live coding*. 4. El Trabajo en redes y colaborativo. En la última sección se presentan las conclusiones particulares a este texto partiendo del análisis de las secciones anteriores (a modo de improvisación final: la deconstrucción, para mantenernos en el lenguaje de la tradición, algunas opiniones personales de los referentes mundiales usados para intuir el futuro del *live coding* se presentan en el anexo A).

3.2. El nacimiento de la idea

Es posible rastrear los orígenes del *live coding* en prácticas artísticas no estrictamente ligadas a las artes computacionales, en otros oficios técnicos y en disciplinas como las matemáticas y la retórica. La historia del *live coding* se intersecta con el nacimiento de las artes algorítmicas y dentro de estas cabe mencionar las composiciones musicales algorítmicas de Guido d'Arezzo en 1026 y otros [21], los juegos de lenguaje o las competiciones matemáticas en el renacimiento, o literatura en novelas como Rayuela de Julio Cortazar [21]. También se pueden mencionar los trabajos de John Cage [37] y sus métodos de composición junto a otros compositores de las vanguardias del siglo 20. En todos estos antecedentes es posible descubrir ideas, conceptos y prácticas ya ligadas a lo que se entiende como *live coding*, es decir, el *live coding* funde sus orígenes con el de las artes algorítmicas.

La idea inicial es crear algoritmos para producir arte, pero cuando hablamos de *live coding*, ¿de qué estamos hablando exactamente?. Como ya se dijo, el *live coding* ha estado en constante movimiento, y de forma bastante flexible se ha transformado, mutado, incluyendo nuevas ideas y prácticas dentro de su estructura [25]. Una de las constantes en su constitución es la idea de ser un acto artístico dado que debe ser "algo" para ser hecho en vivo, y tal vez por esto, ha estado atado a la idea de improvisación [24], la escenificación y las artes escénicas. Por otra parte, al ser algorítmico, su desarrollo ha estado ligado a la evolución de los procesos de cómputo y al desarrollo de nuevos sistemas de procesamiento. Es importante notar en este punto que, una vertiente en el trabajo con *live coding* ha explorado ampliamente el uso de algoritmos no computacionales, podríamos llamarlos cotidianos, es decir, prácticas donde los algoritmos se construyen y modifican a modo de instrucciones en lenguaje ordinario, donde le código (la instrucción) es ejecutado no por computadores sino por personas.

Siguiendo la etimología de la palabra *programar*, del griego prográphein, que es la actividad de escribir en público [35], en el año 2000 Alex McLean y Adrian Ward formaron el dueto SLUB [24], con la intención de hacer música usando sus laptops como instrumentos y escribiendo su propio software. Una de las características más importantes de su trabajo fue el hecho de proyectar sus pantallas para que el público viera el acto de escribir código (en principio sólo comandos de la terminal) como parte del performance-concierto, esta idea evolucionaría hasta el punto en el que es posible modificar el código fuente en tiempo real. Por otro lado, James McCartney había liberado SuperCollider 2 [45] que es uno de los lenguajes de dominio especifico (DSL) pioneros en el desarrollo de un intérprete en tiempo real, junto a un motor de audio capaz de modificar la síntesis y los procesos algorítmicos en vivo. Este lenguaje incluyo una habilidad que según

Rohrhuber [24] pasó casi desapercibida: se podían interpretar nuevas líneas de código aún mientras el motor de audio estuviera produciendo síntesis sin producir una pausa o cortes en el sonido (al menos en la mayoría de los casos). De esta forma y usando esta nueva habilidad del lenguaje, Julian Rohrhuber empezaba su trabajo como pionero del *live coding*, trabajo que lo llevaría a desarrollar la librería jitlib (Just In Time Library), para hacer más fácil la escritura de patrones rítmicos en SuperCollider [63] y que aún actualmente es quizás la librería más usada para el *live coding*.

Además de SuperCollider, estos primeros años ven el nacimiento de varios lenguajes y motores para la producción de síntesis con posibilidades de tiempo real como Nyquist [26] y con un enfoque diferente Miller Puckette [57], en IRCAM, trabajaba en MAX y PureData como lenguajes de flujos gráficos, con los que se podía manipular el motor de audio en tiempo real. El paradigma de estas dos herramientas es completamente distinto a los lenguajes basados en líneas de texto. De forma paralela, en la Universidad de Princeton, Ge Wang y Perry Cook [73] estaban trabajando en Chuck, un DSL con características ya propias al live coding: *"on-the-fly programming"*, procesos concurrentes de forma nativa y controles simples para modificar el código sin parar el motor de audio, chucK, además, incluye un visualizador, el Audicle, que permite la visualización de cada parte del código como entes orgánicos particulares. Aunque este desarrollo nunca pasó de ser una versión de prueba altamente inestable, permite ejemplificar uno de los puntos importantes dentro del *live coding*, esto es, ver el código como una entidad, característica que eleva su rango al nivel del artista, por lo tanto, en el *live coding*, va a ser tan importante el artista como su obra, en este caso el código. Cabe notar que, los anteriores, no son los únicos y además que, estos lenguajes y entornos hacen parte de una larga tradición en la creación de herramientas para el desarrollo de síntesis, composición y música por computador que puede ser estudiada a fondo en [38] y [59] y que están en la familia de los "MUSIC-N languages".

Luego de estos primeros años de exploración y desarrollos iniciales es necesario añadir elementos para una definición más estricta, y en este punto aparecerán dos posiciones, una blanda y una fuerte. La blanda incluye en el *live coding* la manipulación paramétrica de algoritmos, no los algoritmos en sí, es decir, una manipulación trivial. La definición fuerte, donde los algoritmos son manipulados, inclusive de forma dramática en vivo [24] o como lo indica Magnusnon [41], el *live coding* es escribir y manipular los algoritmos en tiempo real. Debe notarse que esta definición hace del *live coding* una herramienta conceptual que permite la disolución de fronteras que el medio de producción artístico mantenía vivas: el muro invisible que divide al artista del espectador, al artista de los medios de producción artística, a la partitura de la composición; el escenario y el auditorio. Todas estas fronteras se van a difuminar en un proceso natural [41] y de naturalización en el proceso de evolución del *live coding*.

Como ya se dijo, no todo algoritmo es computacional por sí mismo, y estas definiciones tienden a dejar por fuera los acercamientos que no hacen uso de computadores o al menos les confieren un lugar secundario que, como se verá más adelante, son un campo amplio y con gran desarrollo dentro de las aplicaciones del concepto *live coding* en la actualidad.

Además de la discusión que plantea, esta perspectiva nos sugiere el lenguaje y las técnicas para transformarlo, modificarlo y extenderlo en vivo, como las herramientas necesarias para entender la práctica del *live coding* y de esta forma acercarnos a prácticas no asociadas con lo digital; mas no por esto dejar de decir con Magnusonn, que el *live coding* tiene un pie en las artes, pero el otro parado firmemente en las ciencias computacionales [41].

El *live coding* ha estado de la mano de los lenguajes de programación, estando presente en los diferentes paradigmas, bien sea la programación orientada a objetos, o paradigmas funcionales, pero básicamente, lenguajes de alto nivel de abstracción. [60]. Eso sí, casi siempre prefiriendo los lenguajes interpretados sobre los

compilados, ya que como el concepto mismo lo propone, es necesario tener la capacidad de transformar los algoritmos sobre la marcha, cosa que no es posible en los lenguajes compilados, al menos no completamente, pero haciendo uso de todas las herramientas disponibles, es posible encontrar lenguajes de texto, o de flujos gráficos y un gran desarrollo dentro de los protocolos de red y de interacción humano computador (HCI).

Esta sección nos da una idea general de cómo el *live coding* fue tomando forma diluyendo otros conceptos y mutándolos en su estructura interna, por ejemplo, el concepto de público o audiencia, que dejará de ser pasiva, entre otros que ya se mencionarán a lo largo del texto. También la programación como ejercicio técnico sufrirá una transformación en su concepción tradicional en general, para llevarla al campo de las acciones artísticas, tanto el programador como artista deben incorporar el error y no solo el error, sino la desnudez de mostrar el error en tiempo real a la audiencia; hacer de esto una fortaleza y parte misma del acto artístico.

Para concluir esta sección es importante resaltar la visión que presenta el *live coding*, permitiendo que el computador, los instrumentos digitales como software e interfaces de interacción o control, y el lenguaje de programación en sí mismo, se transformen en instrumentos musicales o en meta-lenguajes musicales con características únicas, llevando la "máquina cibernética" [39] a convertirse en una herramienta de pensamiento y creatividad con características únicas.

3.3. Diferentes caminos

3.3.1. La pedagogía del código

Para hablar de la pedagogía detrás de los conceptos lógicos, las abstracciones y las estructuras del código es necesario mencionar a Papert [53] y la evolución de este campo desde los primeros trabajos con el lenguaje de programación de propósito general Logo [19]. Estos se presentan como grandes influencias y como un claro precedente al trabajo que se expondrá con respecto a los esfuerzos educativos desde el *live coding* tanto con niños como con adultos, programadores expertos y novatos.

El *live coding* posee una característica que lo hace muy llamativo para el no iniciado, es decir, para aquel usuario de software que nunca ha programado, esto es, la respuesta inmediata de su código como un producto sonoro o visual instantáneo. Esta característica se manifiesta para el conocedor como una gran capacidad de generar prototipos de forma rápida y llevada al extremo, como una posibilidad de usar el lenguaje de programación computacional como una interfaz capaz de ser explotada mediante la improvisación.

Este modelo constructivista, otorga una gratificación al usuario de cualquier nivel [22] y puede verse reflejada en un entendimiento de las relaciones entre el código y los resultados, es decir, los estudiantes-usuarios pueden entender fácilmente las relaciones causa y efecto [65] que no son tan fáciles de observar y tienden a ser más abstractas en la programación con lenguajes compilados por la brecha temporal que introducen en el proceso de transformación del código como instrucciones en texto a resultados tangibles, bien sean visibles o audibles.

Dado que el *live coding* puede plantear visiones alternativas como respuesta al problema de la enseñanza de la programación, el público objetivo para el aprendizaje es extremadamente diverso y el universo de trasfondos, edades, habilidades y demás características es muy amplio; desde niños hasta adultos, programadores expertos sin conocimientos musicales y músicos profesionales sin conocimientos de programación [22] [6], esto gracias a los nuevos modelos de lenguajes de dominio especifico (DSL) donde las abstracciones computacionales desarrolladas son tan sofisticadas que pueden servir tanto al no programador como al experto en sus procesos de creación, prototipado o improvisación [7]. También existen librerías y herramientas livianas pa-

ra adaptar lenguajes tradicionales a las necesidades del live coding, estas se analizarán en la siguiente sección.

De esto, se puede resaltar junto a Aaron que es posible tratar el acto educativo como un performance [6], dado que los objetivos de estos dos actos no están en conflicto y son más bien complementarios. Puede parecer poco clara la relación entre estas disciplinas, pero los principios musicales, bien sea, los perceptivos o los cognitivos son esencialmente abstracciones que pueden ser entendidas como patrones matemáticos [7], por tanto, la notación musical expresada de esta forma es quizás una de las más apropiadas para el entendimiento de su naturaleza funcional. Un gran ejemplo de este tipo de complementariedad es TidalCycles[49], mini lenguaje que por su naturaleza funcional y manejo de patrones se adapta perfectamente a las abstracciones necesarias para la creación de música electrónica bailable.

Una nueva definición del *live coding* aparece en este punto, según Rubin [64], el *live coding* es el proceso de diseñar e implementar un proyecto frente a un curso durante el periodo académico. Es claro que esta definición hace énfasis en el trabajo del aula, aunque la efectividad real de usar *live coding* como guía para cursos introductorios de programación aún no ha sido ampliamente estudiada [31], debemos mencionar un par de herramientas diseñadas para este fin. La primera es Scratch[1], que es un lenguaje de programación por bloques, con orientación infantil y pedagógica. La otro herramienta, que ha sido usada extensamente gracias a su diseño y forma de distribución para el trabajo en la escuela, inserta en el currículo escolar es Sonic Pi[2] [6][10][8].

Sonic Pi es un software de código abierto, así mismo, es una plataforma de desarrollo diseñada originalmente para los mini computadores Raspberry PI1[6]. Este lenguaje fue diseñado para permitir el aprendizaje de la programación de computadores a través del uso de la música. Está orientado al *live coding* y puede ser usado también en *performances* por programadores expertos [6]. Esta es quizá una de las iniciativas más reconocidas a nivel mundial en el campo del *live coding* en general y el *live coding* en la educación, pero existen otras como ixi lang, que hace parte de la suite ixi y es el proyecto de Thor Magnusson [40], Dave Griffiths y el laboratorio transdiciplinario FoAM [3], el trabajo de Rohrhuber [63] y los workshops de SuperCollider, McLean y su enfoque pedagógico asociado a los patrones en los tejidos [47] y la textilidad del código, Blackwell, McLean et al. [13], entre otros, que son referenciados en estos artículos ya mencionados.

Otros trabajos en este campo educativo y que deben ser mencionados son los de Cardenas [25] y los workshops para novatos en TidalCycles y live coding en general al rededor del mundo , así como Betancur [12] con el desarrollo de lenguajes simples o de primera experiencia para niños, campo en el que no hay grandes exploraciones y el trabajo de Herrera, Lobato et al. [33] y el *live coding* para todos enfocado en la popularización de los lenguajes de programación como (e insertas en otras) formas artísticas.

En conclusión, el *live coding* se presenta como una gran herramienta para el trabajo, no solo artístico y escénico a nivel profesional, sino como una respuesta alternativa para la educación y la pedagogía del código.

3.3.2. Tendencias y Usos

El *live coding* se ha infiltrado hábilmente gracias a su naturaleza mutable y alta capacidad de adaptación en diversos campos de trabajo dentro de las ciencias humanas y las ciencias computacionales, por ejemplo y como se mencionó en la sección anterior, su uso en la pedagogía y el trabajo en el aula ha sido ya

[1] https://scratch.mit.edu/
[2] http://sonic-pi.net/
[3] https://fo.am/kernow/

explorado, pero no se ha tocado aún el punto que es tal vez la bandera por la que es reconocido en su forma más amplia: el *live coding* y su uso en las artes, desde la música (en general la música electrónica bailable (EDM)), pasando por el teatro, la danza y las artes escénicas, hasta llegar a las artes visuales y plásticas.

Para iniciar esta sección, se presenta una nueva definición de lo puede ser el *live coding*, pero esta vez en la voz de McLean y Collins [23], donde el *live coding* es una actividad que genera algoritmos en tiempo real para producir EDM. Es necesario dar un salto para introducir un nuevo término y esposarlo al *live coding* para el resto de esta sección, "Algorave". Siguiendo con [23] un Algorave es una actividad donde los algoritmos son explorados en alianza con la producción de música para el baile. Es de notar que las dos definiciones están tan cerca que es posible llegar a confundirlas, si no se aclaran los detalles detrás de ambos conceptos; de todas formas, puede decirse de una vez que, el *live coding* se presta como medio expresivo a la producción de EDM (en este caso específico) para transformar una fiesta tradicional "rave" en un evento trans-formado donde la academia, la diversión, el código y la música dance a todo volumen se entrelazan para formar un concepto: el Rave Algorítmico o Algorave.

Ahora, no todo Algorave está constituido plenamente de *live coding*, no en el sentido fuerte de la definición, ya que es posible encontrar sintetizadores preconstruidos, tanto digitales como análogos y otros tipos de expresiones propias al "rave", pero el *live coding* si ha usado el Algorave como una plataforma de exploración con el público siempre que se ha podido. Y es que, en las condiciones de trabajo y variables de un Algorave, por ejemplo, la interacción directa con un público dispuesto (al menos a ver), el *live coding* se puede mostrar en forma completa y abierta, saltar desde la creación de música generativa, hasta el uso de material sonoro pregrabado, el cual puede ser manipulado en vivo por medio de la escritura y modificación de algoritmos.

Algunos de los lenguajes y entornos ya mencionados han llevado una carrera exitosa en el contexto del Algorave, y en esta lista tenemos a SuperCollider, con el uso de jitlib[24]. PureData y Chuck [72] [74] de manera paralela y como iniciativas independientes, de forma menos extendida. También Sonic Pi, siendo un lenguaje diseñado con fines educativos, ha hecho carrera; e ixi lang, que se puede ver como una extensión parasitaria de SuperCollider, justamente para la creación de patrones según Magnusson [40]. Esta lista es solo un bosquejo del ecosistema de lenguajes y entornos usados. Por sus cualidades y estilo sintáctico, o por su diseño y uso común, los lenguajes más utilizados para el Algorave o los que fueron diseñados para este fin se mostrarán brevemente a continuación:

- Overtone [7] [6]. Se define como un entorno de audio de código abierto, diseñado para la composición musical, pero pensando en usuarios profesionales, bien sean artistas o programadores. Está construido con dos componentes diferenciados, el motor de audio que usa SuperCollider y el intérprete diseñado sobre Clojure que a su vez es un dialecto de Lisp, uno de los primeros lenguajes de programación desarrollados y que sirvió de inspiración para algunos de los lenguajes de paradigma funcional que se usan en la actualidad.

- Gibber[61]. Es un entorno de desarrollo para exploradores Web para la creación audiovisual y composición musical; está desarrollado en Javascript y no requiere de adiciones sintácticas, lo que permite que su código sea ejecutado desde cualquier navegador usando Web audio API. Una de sus mayores fortalezas es que puede ser usado para performances en red, donde múltiples usuarios pueden manipular el código simultáneamente.

- Tidal Cycles (Tidal)[49] [46]. Es un lenguaje diseñado para la creación y manipulación de patrones, dándole prioridad a la velocidad de escritura mas que en la curva de aprendizaje, además tiene un alto enfoque para la ejecución en vivo . Está embebido en Haskell, que a su vez es un lenguaje de

programación de paradigma puramente funcional. Su motor de audio estaba originalmente escrito en C y tenía por nombre Dirt, en las versiones actuales lo portaron a SuperCollider usando el nombre SuperDirt.

- Extempore (también ver Impromptu [69]). Es un lenguaje y entorno de programación, donde el programador está ante un sistema denominado "cyberphysical", esto según Sorensen [16]. Su sintaxis está parcialmente basada en Lisp siendo cercana al dialecto Scheme.
- FoxDot de Kirkbride [36] es una opción para los conocedores de Python, fue desarrollado en 2015 como una librería, donde es Python el intérprete y SuperCollider el motor de audio.
- Conductive como librería también para Haskell de Bell [11] es una librería, donde el motor de audio es hsc3, el cliente de Haskell para scsynth – SuperCollider.

Lenguaje	Estilo/Familia	Autor	Usos
SuperCollider	Smalltalk	James McCartney y otros	Patrones usando JITLIB, Síntesis, Análisis, Composición algorítmica
PureData MAX/MSP	Dataflow	Miller Puckette	Síntesis, Análisis, Composición algorítmica, Gráficos usando GEM, Extendible
Tidal	Haskell	Alex McLean y otros	EDM, Manipulación de patrones, Loops
Sonic PI	Ruby	Samuel Aaron y otros	Educación, EDM
Chuck	C++	Ge Wang y otros	Síntesis, Análisis, Composición algorítmica, Extendible
Gibber	Javascript	Charlie Roberts	Síntesis, Patrones, Colaborativo, Web
Overtone	Clojure	Samuel Aaron y otros	Síntesis, Manipulación de muestras, Colaborativo
Extempore Impromptu	Scheme	Andrew Sorensen	Computación Cyber física, Porcesamiento de señales al vuelo
FoxDot	Python	Ryan Kirkbride	Patrones e Instrumentos definidos en SC
Conductive	Haskell	Renick Bell	EDM, Manipulación de Patrones

Una lista más completa puede verse en línea [1] (revisar anexo C), donde también es posible rastrear lenguajes para la visualización de gráficos tanto vectoriales como 3D, o manejo de contenido pregrabado como

videos, pero se mencionan algunos como kodeLife, Cyril, VVVV y Fluxus y livecodelab, por citar algunos.

Además de esta fuerte tendencia, es posible rastrear aplicaciones donde el *live coding* es usado como medio expresivo, entre las que cabe mencionar el *live coding* no computacional o computación no convencional, descrita en [33], los trabajos con baile y coreográficos de Sicchio [48], y otros que pueden verse en [21] donde el código se usa de formas no convencionales. Para ver algunos de los precedentes y trabajos representativos latinoamericanos en especial en México, ver el trabajo de Rodriguez que resume los precedentes, incluyendo festivales, artistas y aproximaciones enmarcadas en el *Mexican style* [62].

Otra aproximación sobre las tendencias y usos del *live coding* hace referencia a la pregunta de si ¿es posible tener un *performance* colectivo donde el medio expresivo sea precisamente el *live coding*?. La respuesta a esta pregunta se puede vislumbrar en algunos de los casos ya expuestos en secciones anteriores, por ejemplo, Gibber [60], es un entorno donde el código puede ser editado de manera colectiva. Esto lleva a intentar una nueva definición de lo que es el *live coding*, caso en el que se podría describir como: el acto de producir algoritmos de forma colectiva, por medio de redes de comunicación. esta definición conduce a los trabajos de Ogborn y la Cybernetic Orchestra [50] [51], Boyle [14], Hindle [34] y las "laptop orchestras": Princeton - PLOrk, Stanford - SLOrk [71], LO2rk (Linux Laptop Orchestra) [17] y Concordia-CLOrk. Aunque es importante notar que el trabajo de estos ensambles no es siempre un trabajo centrado en el uso del *live coding* donde también se usan otro tipo de interfaces para sus actos en vivo.

En este punto, es necesario hablar sobre la portabilidad de las herramientas del trabajo colaborativo, como el uso extendido del protocolo OSC [75], el cual permite interconectar interfaces, dispositivos y lenguajes diversos para facilitar el ensamble, por medio de un protocolo bastante simple de implementar en la mayoría de las herramientas ya mencionadas. También es posible usar redes mediante mensajes TCP/IP (protocolo estandar de comunicación por redes) y lograr sincronizar los relojes de cada máquina dentro del *performance*, solución ya planteada por Collins et all. [24]; o trabajar desde una plataforma neutral, donde se pueda evaluar código sin instalar paquetes extra, como en el caso de Extramuros [51] y Estuary [52], donde todo el performance puede ser centralizado en un servidor, mientras los otros participantes envían su código en forma de mensajes. Esta solución particular es de gran importancia para los contextos pedagógicos donde la instalación no es justamente el punto central ni el objetivo.

Debe resaltarse que este tipo de entornos, que no están orientados a un lenguaje particular, es decir, que son neutrales en su concepción, además de ser una potente herramienta pedagógica por su portabilidad y por su fácil y rápida adaptación a cualquier lenguaje o sistema, permiten la expansión del *live coding* a campos donde el ensamble colectivo y la construcción colaborativa son vitales, permitiendo que, nuevos usuarios con distintos niveles de conocimiento, puedan generar alternativas de trabajo en red.

Esta neutralidad es importante para el concepto general del *live coding*, ya que da libertad a los programadores de escoger y diseñar las herramientas que mejor se adapten a su forma de ver el mundo, mas no por esto, aislarlos de la comunidad general; por el contrario, asimilándolos y dándoles cabida en el trabajo general.

3.4. Conclusiones

En este capítulo se ha presentado un recorrido por el *live coding* y su historia reciente, como concepto independiente, por lo que una de las cosas más importantes que logra es reunir en un solo documento los referentes teóricos y artísticos de la práctica del *live coding*, desde sus tendencias dominantes, pasando por el

trabajo conceptual y pedagógico, llegando a las corrientes experimentales y no tan conocidas. Por lo tanto, puede servir como referente para trabajos posteriores y para la consulta de investigadores, también como guía para nuevos live coders interesados en elegir un camino, o como referencia de trabajo en el desarrollo de propuestas de investigación-creación, en los campos de la programación creativa o cualquiera de los demás campos que se han mencionado.

La naturalización técnica y la posterior adaptación dentro de otros campos de investigación y desarrollo artístico, puede verse por la capacidad del *live coding* para integrarse en nuevos medios de expresión y mutar fácilmente para resolver necesidades complejas; por lo que el término como tal, puede llegar a ser asimilado de forma gradual en el quehacer cotidiano de otras disciplinas artísticas, ciencias, o cualquier tipo de trabajo colaborativo, desarrollado en contextos de la improvisación o la resolución rápida de problemas.

El *live coding*, por lo tanto, no se presenta como una técnica novedosa para ser usada por fuera de los contextos tradicionales, sino mas bien, como una renovación de conceptos explorados desde la antigüedad, pero adaptados a las nuevas tecnologías disponibles y a la gran capacidad de cómputo con la que se cuenta en la actualidad. Teniendo esto en cuenta, el desarrollo y evolución de la práctica del *live coding* seguirá transformándose y absorbiendo de manera continuada, bien sea bajo su nombre actual, como arte algorítmico, o de cualquier forma que el mismo concepto deba tomar, para mantener su cualidad adaptativa y empatía formal, como medio de expresión humano.

La pedagogía del código se propone en este capítulo, como un concepto a ser profundizado de forma consiente, dado que, aunque hay un trabajo que se viene realizando en el área, todavía se plantea como un problema técnico más que conceptual, no hay un público definido claramente, lo que evidencia que el campo está abierto a la exploración y discusión.

Una última definición se puede intentar, siendo esta la que ayudará a concluir esta revisión y tal vez abra un horizonte en la posible naturalización y generalización de la práctica del *live coding*: el *live coding* es una práctica que engloba toda actividad, donde los algoritmos, (entendidos como cualquier tipo de instrucción organizada en pasos-definiciones-alteraciones) y la posibilidad de alterarlos, modificarlos o cambiarlos totalmente en el acto (en vivo) son el material esencial. Con esta última definición se transforma el *live coding* en una herramienta para la expresión y por lo tanto para la comunicación.

Historia II - El enlazador

El paradigma de programación imperante en la actualidad, describe los métodos y funciones a realizar inscritas dentro de objetos; estos objetos, entendidos como arquetipos, como modelos ideales de cualidades, variables y constantes, que determinan el comportamiento de familias completas, de espacios de memoria que, luego de ser procesados y modificados, pueden ser de nuevo llamados por su nombre propio, para que revelen sus secretos visibles o develen bajo ciertas circunstancias sus cualidades mas profundas. Es el programador el que, como Dédalo, llena de vida el artificio, el que usando la techné y su propio envoltorio, puede transformar el sueño en vuelo, y la historia en alquimia.

4. Tres necesidades, tres procesos, tres interfaces

En este capítulo se narrará, a modo de tema y variaciones, el proceso desde la concepción inicial hasta la implementación y uso, de tres diferentes sistemas desarrollados en el marco artístico, en tres diferentes grupos de trabajo. El primero, con el colectivo **Algo0ritmos**, dirigido al desarrollo de procesos de composición musical con algoritmos escritos y modificados al vuelo, usando los principios del *live coding* . El segundo, con la **Cybernetic Orchestra**, enfocado en el trabajo pedagógico con programadores no experimentados (inicialmente con niños) y el desarrollo de herramientas para la generación de experiencias significativas en el contacto inicial con los lenguajes de programación, pero extensible a la composición en red y ensambles colaborativos, el tercero, con el colectivo **CICLUX** y los semilleros de investigación creación del ITM, **ACORDE** y **Cinevivo**, un sistema de interfaces para la creación y ejecución de piezas audiovisuales en vivo, usando narrativas experimentales, que incluyen lenguaje de programación, interfaces físicas e interfaces gráficas de usuario (GUI).

Teniendo en cuenta las necesidades, el tiempo disponible, el conocimiento previo y trasfondo de los colectivos y sus integrantes se procedió a asignar una herramienta. Este proceso no se llevo a cabo de forma previa o planeada, sino mas bien, en el pasar de los días y según el contexto y urgencias de que la obra planteaba.
Como ya se mencionó en la introducción, esta tesis usará la investigación-creación como excusa para abordar tanto el trabajo teórico como la creación artística misma y se valdrá de esta misma metodología para brindar como parte de la obra misma el proceso de formalización y escritura, haciendo de este una reflexión que se enmarca y fundamenta en los mismos principios que las obras e interfaces producidas en el proceso de esta tesis [5].

4.1. Variación I

En el marco de la construcción de un programa para la enseñanza de la música electrónica a un grupo diverso de personas, durante los procesos de implementación del Programa *"Medellín vive la música"* en el 2014, se reunió a un grupo de expertos para crear un comité de trabajo, desde el cual debería emerger un programa curricular que incluyera las herramientas de los nuevos medios en la enseñanza de la música. De este experimento nace el colectivo Algo0ritmos.

Algo0Ritmos es un colectivo de conformación libre, lo que implica que cualquier persona interesada en aprender sobre código, programación, algoritmos, música electrónica o por simple curiosidad; es bienvenida y aceptada de forma inmediata. El colectivo nace en el año 2014, dentro de un programa financiado con recursos públicos de la Alcaldía de Medellín, junto a otras organizaciones que se sumaron al proyecto como MOVA y el Colaboratorio del Parque Explora. La idea fundacional era simple, tomando el manifiesto del Algorave, se pretendía la creación de talleres abiertos a cualquier curioso. Por lo tanto, la necesidad inicial era buscar un lenguaje de programación que permitiera el prototipado rápido de objetos musicales que, de una u otra forma, fueran bailables o al menos incitaran al movimiento. También era necesario que al mismo tiempo el lenguaje de programación tuviera una sintaxis que luego pudiera ser aplicada a otros contextos no

necesariamente musicales. Fue bajo esta premisa, que el colectivo decidió escoger chucK como lenguaje base para el desarrollo de una nueva librería, Chmusick, que satisficiera las anteriores necesidades.
Durante los talleres los usuarios fueron los que con sus preguntas, errores y experimentos lograron poner a prueba el trabajo de desarrollo de la librería. Cada cambio en la sintaxis o el desarrollo de los métodos y miembros de cada objeto de la librería nacieron en este contexto, donde el trabajo se hacía de forma práctica. Posteriormente, se evaluaban en grupo los avances y problemas que se hubieran presentado en la cotidianidad. Luego de esto se discutían las posibles soluciones o enfoques que podrían tomarse para enfrentar los problemas, para luego ser implementados.

Uno de los puntos claves, y la mayor necesidad para el desarrollo de una librería de este tipo, es que para garantizar una mayor velocidad a la hora de escribir hay que sacrificar velocidad en la curva de aprendizaje y viceversa. Lo anterior obliga a poner en una balanza constante el trabajo, dado que el colectivo artístico que requería de una solución se movía entre Algorave y la pedagogía. Cabe notar que, justamente es ésta ambigüedad la que permite una gran variedad de perfiles en la configuración del grupo, dado que era posible introducir todo tipo de nuevos enfoques en la forma de resolver los problemas.

A continuación se describirán algunos aspectos relevantes de la librería Chmusick, necesarios para entenderla como una interfaz musical.

4.1.1. Chmusick

Chmusick es actualmente una librería dividida en dos modelos sintácticos diferentes. El original está escrito usando el modelo mismo del lenguaje que se extendió, es decir, chucK, y como tal es una colección de objetos con características propias a las abstracciones musicales, que pueden ser manipulados en tiempo real o al vuelo (*on-the-fly*). Además, gracias a esta habilidad y a un algoritmo extra implementado para garantizar la sincronización rítmica necesaria en el EDM, la librería puede ser usada en contextos educativos o como herramienta expresiva para programadores mas avanzados. Este modelo está en la familia de chucK mismo y de otros DSL como SonicPi.

El segundo modelo está pensado de forma cercana a los paradigmas funcionales, al menos en la sintaxis, aunque no por esto deja de ser un lenguaje orientado a la creación de objetos. Desde ese punto de vista, Chmusick es un pariente cercano a ixilang y a Tidal [40] por el tipo secuenciador de pasos, ya que usa la manipulación de patrones para lograr cambios en la música [46].

Para entender las diferencias veamos algunos ejemplos de código, que permitan ver las diferencias y similitudes con otros DSL, y al mismo tiempo reconocer el proceso de construcción como un trabajo colaborativo, donde cada nueva adición y/o corrección de la librería se realizaba para resolver problemas que alguno de los usuarios había presentado en su trabajo personal con ella.

Para empezar se mostrarán tres ejemplos donde se pueden ver las diferencias entre los dos modelos sintácticos, Chmusick (1v) y Chmusick (2v) y chucK, así mismo servirán para explicar los detalles internos de la libería.

Chmusick (1v)

```
Drum base => dac;
base.drum([1,0],[0,1]);
```

Chmusick (2v)

```
Chmusick live => dac;

live.file("bd") => Buffer.d1.read;
live.file("hh") => Buffer.d2.read;

live.play(Buffer.d1,live.every(2));
live.play(Buffer.d1,live.rotate(live.every(2)));
```

chucK

```
SndBuf bd => dac;
SndBuf hh => dac;
me.dir() + "bassDrum.wav" => bd.read;
me.dir() + "hitHat.wav" => hh.read;
60/120.0 => float tempo;
while(true){
    0 => bd.pos;
    (tempo/2)::second => now;
    0 => hh.pos;
    (tempo/2)::second => now;
}
```

En esta comparación es posible ver cómo la sintaxis de ha sido elevada a un nivel más alto en el primer caso (Chmusick (1v)), dejando los detalles ocultos en la librería misma. Lo anterior permite que el programador trabaje desde su expresividad, sin preocuparse demasiado por detalles técnicos. El segundo código (Chmusick (2v)) está en un nivel intermedio, dado que es posible controlar detalles más precisos como la muestra específica que se quiere usar, pero sin ser tan profundo como la sintaxis cruda. El último segmento de código está escrito en la sintaxis cruda de chucK, y se puede observar cómo Chmusick simplifica la creación de los objetos e instrucciones. También es posible observar que el primer modelo sintáctico esgrime su parecido a los secuenciadores de paso tradicionales, el segundo, el trabajo con funciones, patrones y transformaciones de patrones.

El primer modelo sintáctico, Chmusick (1v), es una abstracción donde los arreglos que se usan como parámetros funcionan como interruptores de encendido y apagado en el patrón. En este caso, ambos patrones son simétricos, pero no es más que una casualidad dado que es posible implementar poli-ritmos y ejecutar patrones de diferentes tamaños.

```
base.drum([1,0],[1,1,1]);
```

El código anterior por ejemplo ejecuta un ritmo ternario contra uno binario, permitiendo la ejecución de la base rítmica de un Mapalé de la costa Caribe colombiana o cualquier ritmo de estas características.

El segundo modelo sintáctico, Chmusick (2v), como ya se dijo, está orientado al uso y trabajo con funciones, por lo tanto, aunque el resultado que produce es el mismo, abre la posibilidad de introducir de forma más profunda conceptos computacionales diferentes, complementando así las carencias del primer modelo sintáctico de la librería. Se puede observar que la función every() permite la creación del mismo arreglo del primer ejemplo, es decir, un arreglo donde los unos son interruptores de encendido y los ceros de apagado. Así mismo hay otras funciones como play(), file() y rotate() que cumplen pequeñas tareas dentro de la composición del algoritmo, pero que claramente actúan en un estilo mas funcional.

La forma del primer modelo sintáctico diferencia cada objeto del lenguaje como una abstracción de un instrumento musical, es decir, Drum es la clase encargada de manipular muestras de tipo percutivo añadiendo además métodos para los *refills* y otros métodos bastante propios a la manera tradicional de pensar la base rítmica. Así mismo es posible crear objetos Armónicos y Melódicos de forma independiente con características que le son propias a las abstracciones musicales representadas en un estilo tradicional, como lo muestra el siguiente ejemplo.

```
Harmony pad => dac;
pad.SinOsc(["Fm","Fm","Fm","Eb"]);
```

Se puede observar cómo el objeto Harmony recibe como parámetro un arreglo de acordes escritos en notación clásica, característica que le es única y que no puede ser usada en objetos de la clase Drum por ejemplo. Así mismo, la librería tiene objetos para generar síntesis de Frecuencia Modulada (FM), Amplitud Modulada (AM) u otros tipos ya clásicos de síntesis digital, que pueden ser usados de forma similar a los ejemplos descritos, es decir, con métodos diferenciados según su naturaleza.

Con un enfoque diferente, como ya se dijo, el segundo modelo sintáctico está orientado a la creación de un solo objeto que, desde su creación, puede ser usado de forma única para el manejo de cualquier tipo de generador de audio, bien sean nativos al lenguaje o desarrollados como nuevas clases, por ejemplo:

```
Chmusick live => dac;
SinOsc bass => dac;
live.file("bd") => Buffer.d1.read;
live.play(Buffer.d1,live.every(2));
live.play(bass,live.every(2,60));
```

En este ejemplo puede verse que el objeto general de control "Chumusik", una vez instanciado con el nombre "live", es usado para la ejecución de muestras de la carpeta "bd", pero al mismo tiempo sirve para la ejecución de un patrón específico usando el generador de audio SinOsc, que es un objeto nativo dentro del lenguaje chuck. Es posible usar este objeto de control único para ejecutar todo tipo de patrones bien sea con muestras o con síntesis.

4.2. Variación II

En la enseñanza de la programación, existe un ejemplo bien conocido en todos los lenguajes, donde es posible ver de forma somera la sintaxis, el paradigma de programación y hasta el estilo que se considera estándar al programar en dicho lenguaje, este ejemplo tradicionalmente recibe el nombre de *"hello world"*.

Al revisar este primer programa, que puede ser ejecutado en un lenguaje de programación cualquiera, es posible ver que en su gran mayoría, los lenguajes generan un texto plano como salida. Esto es distinto con los DSL, donde la salida puede ser un elemento gráfico o una onda sinusoidal simple en una sola frecuencia, este es el caso de chucK.

Ahora, aunque este primer ejemplo puede ser ejecutado con pocas lineas de código, es poco interesante como material expresivo, y en general el tiempo entre la ejecución de este programa y un desarrollo más complejo, es considerablemente alto. Es decir, los lenguajes de programación tienen una curva de aprendizaje

con una pendiente bastante alta entre un primer ejemplo ejecutable y un desarrollo con expresividad artística.

En el trabajo pedagógico, este pequeño detalle es de gran importancia ya que esta primera experiencia determina en gran medida el interés posterior en el aprendizaje del lenguaje. Esto es evidente en casos donde el aprendiz no viene al aprendizaje con otras necesidades, sino mas bien impulsado por la curiosidad.

Es bajo estas premisas que nace CQenze, como un mini lenguaje diseñado para brindar una primera experiencia programando, altamente significativa y expresiva, así el resultado sea caótico, o aleatorio, en cualquier caso mantiene el precepto de la velocidad en la obtención de la experiencia sacrificando el aprendizaje de conceptos computacionales o musicales más profundos.

La respuesta, conceptualmente hablando, es simple, elevar las abstracciones a un punto en el que el mapeo entre el texto/sintaxis y su correlación con la respuesta de la salida sea lo mas cercana posible. Una relación uno a uno entre símbolo y resultado.

Es por esta razón por la que CQenze se convierte en la herramienta ideal para demostrar la relación texto-proceso, que se produce en la programación, en especial para personas que no tienen ninguna experiencia (EUP). De esa forma, al producir un resultado instantáneo y evidente [9] no se necesita de capacitación previa y tiene una alta capacidad de impacto, sobretodo en la primera experiencia.
Ahora, ¿qué es CQenze? CQenze es una capa sintáctico super impuesta sobre chucK, inspirada en otros lenguajes de programación como TidalCycles[46] e ixilang [40], y claramente puede verse como una analogía a los secuenciadores de pasos, bien sean análogos o digitales. Esta orientado a la modificación de patrones en tiempo real e intenta ser lo suficientemente simple e intuitivo como para servir en un primer enfrentamiento a los lenguajes de programación, por lo que puede considerarse un lenguaje de primera experiencia y esto lo hace ideal para el trabajo con programadores novatos[33].

Una consecuencia de que CQenze esté implementado como lenguaje para una primera experiencia en programación, es su funcionalidad basada en Extramuros [51], que es un entorno para compartir buffers de texto, de forma que no requiera instalación y pueda ser probado con un servidor externo. Lo anterior permite que el código sea escrito y lanzado desde el explorador web, usando cualquier dispositivo con acceso a internet o a una red local. En ese caso sólo el servidor debe tener chucK y el interprete de CQenze.

Como ya se dijo, CQenze es una capa sintáctica sobre chucK, sobre el cual actúa como parásito, usando el motor de síntesis, y las capacidades específicas que lo hacen tan potente: ejecución en tiempo real (real time) y su famoso “spork ” (ejecución paralela). CQenze a diferencia de Chmusick oculta su naturaleza, y para el usuario no es necesario tener ninguna relación directa con chucK.

Es necesario decir que CQenze es autónomo como interfaz y como lenguaje para el live coding, pero la producción de audio depende enteramente del lenguaje que lo hospeda, es decir, chucK. La anterior conlleva a que CQenze tenga todos los componentes necesarios para analizar la sintaxis, generar mensajes de error propios, su propio editor de texto y un interprete independiente.

Por estas cualidades, este lenguaje fue escogido para ser integrado en Estuary [52] que es una plataforma para la colaboración y ensambles en redes donde personas con experiencia diversa pueden contribuir en la creación de música electrónica, usando lenguajes de programación y técnicas del live coding. Esta plataforma es el proyecto de investigación del Networked Imagination Laboratory (NIL) junto a la Cybernetic Orchestra.

4.2.1. Modelo Sintáctico de CQenze

CQenze acepta comentarios de línea usando la sintaxis tipo C "//", y frases válidas que deben tener la estructura *objeto* + *operador* + *vector* + ";". En caso de no ser válida, el interprete genera cerca de diez mensajes de error diferentes, según sea el caso.

El modelo sintáctico de CQenze usa arreglos (para este caso es posible decir listas) de símbolos (+ ó -), donde cada símbolo actúa como un interruptor (activado, desactivado) de eventos que tienen por duración un ciclo dividido por el número de eventos presentes en la lista. Es decir, el arreglo funciona como un secuenciador de pasos o *drum machine*, haciendo una referencia directa a la abstracción que se pretende representar [46] .

El operador de asignación puede ser "=" ó ":", cada uno con una funcionalidad temporal distinta. El operador ":" distribuye los eventos forzando las muestras a ser reproducidos en eventos simétricos durante la cantidad de ciclos determinados (por defecto dos). Es decir, cada evento programado tendrá la misma cantidad de tiempo que los demás en la misma lista, y este tiempo de ejecución será relativo a la cantidad de eventos. El operador "=" corta las muestras a negras -en sentido musical- (quarter notes, un futuro desarrollo es el uso de diferentes subdivisiones) y fue implementado con la intención de usar notación convencional.

Ahora, los eventos se programan en una lista compuesta por los símbolos "+" y "-" lo que indica activado/desactivado, cerrando cada linea de código por un punto y coma ";" o por la inserción de una nueva linea. De esta forma, se presenta como un mini lenguaje más "gráfico", en el sentido que es posible para el programador visualizar la concurrencia de los eventos con sólo ver el código.

De esta forma, el mismo efecto generado por los ejemplos ya mencionados en chucK y Chmusick puede escribirse en CQenze así:

```
tempo 120;
bd:+-;
hh:-+;
```

Se puede observar la simplicidad y facilidad de la sintaxis con respecto a Chmusick, y a chucK mismo.

4.3. Variación III

El artista audiovisual se encuentra en un punto donde su interacción con la tecnología y su obra están tan íntimamente ligados, que las fronteras entre las disciplinas artísticas se han ido borrando incorporando incluso nuevos profesionales en su ejercicio creativo.

Cinevivo es un semillero de investigación creación que reúne estudiantes y docentes de diferentes facultades y carreras interesados en el desarrollo de contenido audiovisual, usando narrativas experimentales. Para este fin, hace uso de herramientas digitales y de elementos electrónicos tanto de punta, como dispositivos obsoletos.

ACORDE es también un semillero de investigación creación del ITM, pero en su caso, este semillero esta orientado a la composición tanto musical como visual, más enfocada a la musica electroacústica y el desarrollo de ensambles para la interpretación de estas piezas en vivo.

Cinevivo pretende buscar nuevas respuestas y formas de creación de piezas en vivo, por esa razón, se sumo a la presente investigación para servir como laboratorio de experimentación y desarrollo, tanto en el uso

de interfaces físicas específicamente diseñadas para las piezas artísticas a ejecutar y con la disposición de experimentar nuevas formas, como en el uso del live coding a modo de interfaz. En el caso de ACORDE, lo que se buscaba era el desarrollo de interfaces que permitieran tanto la creación como la ejecución de piezas electroacústicas y visuales en contexto de concierto. Cinevivo sin embargo, tenía un calendario apretado con presentaciones tanto nacionales como internacionales, y compromisos académicos constantes, lo que generó un reto constante para dar respuesta a problemas específicos dentro de su proceso creativo.

Para entender mejor el proceso es necesario decir que el trabajo del semillero Cinevivo se desarrolla en el campo del live cinema, el cual es un concepto audiovisual que está construido en torno a varios puntos que pueden ser listados y resumidos de la siguiente forma:

- El espacio que se entiende como el lugar compartido entre el artista, sus dispositivos, las proyecciones y el público.
- El tiempo entendido como la oportunidad de la improvisación y la interacción entre el performer y el público por medio de las proyecciones y el sonido.
- El loop es un recurso en el que se repite una misma secuencia visual con la idea de romper la estructura narrativa tradicional.
- La proyección es el medio por el cual el artista y el público pueden entablar una relación directa, es la visualización del proceso creativo.
- El performance es el resultado del acto en vivo de artistas y músicos involucrando el cuerpo y el movimiento en el espacio.
- En el Live Cinema entendemos al espectador no como un sujeto contemplativo, sino como un actor que puede intervenir, ser creador y creación [43][58].

4.3.1. El Software CineVivo

A partir de las anteriores premisas se crea el software CineVivo, como una herramienta para la creación colaborativa de live cinema. Con esto en mente se puede decir que CineVivo es un software escrito en C++ usando toda la potencia y estabilidad de OpenFrameworks (OF).

OpenFrameworks es un librería de C/C++ de código abierto, especialmente escrita para hacer piezas creativas, y por lo tanto sus principales pilares son el audio y el video. También esta diseñada para controlar instalaciones artísticas por medio de sensores y microprocesadores, es decir que su código puede funcionar desde Arduinos hasta RaspberryPi o computadores con sistemas operativos tradicionales, ya que soporta Windows, Linux y MacOS.

Una de los fortalezas de esta librería es la uniformidad de su sintaxis, que la hace fácil de comprender en comparación con los lenguajes crudos, o librerías no estandarizadas [70]. Al mismo tiempo tiene control del bajo nivel, por estar escrita sobre C/C++, lo que le da todo el potencial necesario para ser una librería robusta y estable.

Open frameworks tiene una gran cantidad de librerías de expansión que pueden ser descargadas desde *http://ofxaddons.com*, con una gran comunidad que le da soporte y mantenimiento constante, lo que hace de este plataforma una herramienta ideal para los procesos pedagógicos y por esta razón se escogió como plataforma para la construcción del software CineVivo.

En CineVivo, cada fuente de video al ingresar se divide pixel a pixel, por lo que el video puede ser granularizado hasta este punto, si es requerido. Para propósitos prácticos, este proceso solo se lleva hasta el punto de sacar tramas de un pixel de ancho por el alto del video original. Así, un video de 300×200 se divide en 300 tramas de 1×200, para ser desplazadas en el espacio de forma independiente. El proceso de color y opacidad se realiza a cada fuente de vídeo por separado, aunque es posible realizarlo pixel a pixel también, donde el límite es la capacidad de hacerlo en tiempo real.

De forma general, el núcleo del software es una matriz donde cada pixel, o trama (conjunto de pixeles), se agrupan bajo un identificador único, permitiendo de esta forma aplicar transformaciones de color, espacio, posición y forma. Además, el software permite usar otros procesos de análisis implementados usando elementos de visión por computador (*Computer Vision*), con la librería openCV, para la síntesis de imagen en tiempo real. Todas las operaciones se hacen de forma independiente, permitiendo que el operador asigne todo tipo de controladores o interfaces, ya sean físicas, GUI, o por texto; para su control y aplicación en vivo.

El software además tiene implementados los protocolos de comunicación MIDI y OSC, lo que le permite interactuar con interfaces físicas diversas. Fue esta propiedad la que permitió el desarrollo de tres interfaces físicas, que sirvieron de controladores para la ejecución en tiempo real de las piezas creadas usando el software. Estas interfaces fueron desarrolladas por el equipo creativo de trabajo; un diseñador industrial (Luis Rodríguez) encargado de la ejecución material y la materialización escultórica de las interfaces, quien trabajó de la mano de los artistas para concretar la elección de materiales y controles para los actos vivo. El desarrollo del hardware y del software corrió a cargo del autor de esta investigación, procurando mantener los principios del colectivo (el reciclaje, la obsolescencia, la ejecución en vivo) e incorporando los conceptos desarrollados en los anteriores capítulos.

El trabajo con ACORDE se baso en la implementación de las mismas interfaces físicas desarrolladas para Cinevivo pero usando el *live coding*, tanto Chmusick, CQenze y el apoyo de otros lenguajes como Processing o PD para complementar el proceso y dar una respuesta a las necesidades especificas de este semillero.

Historia III - Lacrimosa

"Hola, ¿ estás ahí ? . . . déjame llorar en paz". Software de Inteligencia Artificial ASH.

5. El impacto de los cambios

A lo largo de este capítulo se mostrará el componente experimental y práctico de esta investigación, que se enmarcó en el desarrollo de nuevas interfaces, en un proceso que unió al artista, a la pieza y al desarrollador, en todas las etapas de depuración de las herramientas técnicas, para llevar a cabo la construcción de piezas artísticas a ser ejecutadas en vivo.

El proceso se apoya además en documentación audiovisual, que se incluye como anexo al presente documento, ya sea en forma de material complementario o enlaces a recursos en internet.

5.1. Metodología

Cómo se ha mencionado previamente, la realización de este trabajo de investigación requirió la participación de colectivos artísticos con necesidades especificas (expuestas en el capítulo anterior), dispuestos a participar tanto en el desarrollo y evaluación de las interfaces, como en su apropiación y uso en actos en vivo, que permitieran observar la relación artistas-obra-interfaz.

Se escogieron dos semilleros de investigación-creación del Instituto Tecnológico Metropolitano de la ciudad de Medellín adscritos a las Facultad de Artes y Humanidades, un tercer grupo formado con los participantes del colectivo artístico Algo0ritmos, y un cuarto grupo con los estudiantes participantes de la Cybernetic Orchestra, agrupación adscrita al *"Networked Imagination Laboratory"* en la Universidad de McMaster, en Hamilton, Ontario, Canadá.

Los grupos estaban constituidos de la siguiente forma:

- **Semillero ACORDE.** Este grupo centra su trabajo en la creación de música electroacústica. Constaba, para el momento de este trabajo, con 4 integrantes con experiencia tanto en la creación de música, como en la programación de computadores con lenguajes de programación diversos, en especial con el lenguaje de flujos gráficos PureData (PD).

  Dos de los integrantes, eran estudiantes de la tecnología en Informática Musical; un tercero, procedente del programa de Artes de la Grabación (ambos programas pertenecientes a la Facultad de Artes del ITM), y el cuarto, un docente con grado de Maestría en Artes.

  Este semillero usa su espacio de trabajo para el desarrollo de música, tanto la ejecución como la composición de piezas electroacústicas, donde se combinan los instrumentos tradicionales con nuevas formas de creación sonora, como los lenguajes de programación.
- **Semillero Cinevivo.** El semillero Cinevivo es un colectivo artístico interesado en el desarrollo de piezas audiovisuales usando narrativas experimentales, nuevas interfaces y por lo tanto, nuevas formas de interacción tanto con el publico como con el *performance* en sí mismo. El número de integrantes que participó en esta investigación fue de seis (6) personas.

Algunos con amplia experiencia en la creación de contenido audiovisual, pero ninguno de ellos familiarizado con la programación. Sólo dos de los participantes dijeron haber programado alguna vez y sólo como parte de algún curso universitario, es decir, ninguno de ellos se considero un programador.

Específicamente, el semillero estaba constituido por cuatro estudiantes de semestres diversos del programa de Artes Visuales de la Facultad de Artes y Humanidades del ITM, junto a dos docentes, uno del programa mencionado y otro del programa de Diseño industrial. Antes de este proyecto, el semillero construía sus piezas usando software comercial para la edición y ejecución del material pregrabado, recogido durante las etapas de preproducción.

- **Colectivo Algo0ritmos.** Este colectivo trabaja con la creación de musica electrónica bailable (EDM), con dos objetivos, la pista de baile y la enseñanza de la programación a programadores no expertos a través del uso de la música. A diferencia de los anteriores, este grupo tiene por naturaleza una inclinación al trabajo con la programación de computadores, en especial con el lenguaje de dominio especifico (DSL) chucK. Sus integrantes afirmaron conocer también otros lenguajes tanto orientados a objetos como funcionales.

  Para la investigación presente, el colectivo tenia 8 integrantes con diferentes antecedentes académicos, edades diversas, pero como ya se dijo, experiencia en la programación, que se puede describir como nivel medio a avanzado.

  El grupo se constituía de la siguiente forma: Un ingeniero de sonido con amplia experiencia en audio digital y programación en lenguajes tradicionales, dos personas provenientes de las ciencias sociales, una socióloga y una bióloga, sin ninguna experiencia en programación, música o artes. Los otros integrantes, todos estudiantes de pregrado de diferentes programas y niveles.

- **Cybernetic Orchestra.** El grupo adscrito a la Universidad McMaster en Hamilton, ON, Canadá centra su trabajo en el uso de las redes informáticas para la creación de ensambles, principalmente usando la técnica del *live coding*, para crear piezas sonoras. Este grupo cuenta con un numero variable de integrantes. Así mismo, aunque todos pertenecen a la universidad, vienen de carreras diversas y hay estudiantes tanto de pregrado, como de maestría y doctorado. Es por esto, que es difícil describir a detalle el grupo mismo, y está característica lo hace también un grupo muy dinámico para el desarrollo de la propuesta artística especifica.

5.1.1. Fase 1: Exploración

Dado que los grupos de trabajo tenían diferentes expectativas y su trabajo estaba orientado sutilmente hacia objetivos distintos, se propuso una fase de exploración previa. Cada grupo de trabajo hizo una exposición de forma natural y espontánea del tipo de pieza que estaban acostumbrados a construir, junto a una pequeña descripción de las herramientas usadas para tal fin. La descripción debía incluir, el tipo de interfaz usada, tipo de herramientas empleadas, un estimativo de la cantidad de tiempo empleado en capacitación y entrenamiento y, siempre que fuera posible, el grupo debía hacer una demostración en vivo.

Se identificó como cada grupo estaba acostumbrado al manejo de ciertas herramientas especificas, por lo que se escogieron herramientas que estuvieran cerca a esa experiencia, para que la introducción controlada de pequeños cambios no fuera una experiencia de transformación completa desde el inicio.

Luego de este trabajo exploratorio inicial se tomo la decisión de distribuir las herramientas en cada grupo de la siguiente forma, aclarando que los procesos no fueron simultaneos y que los cambios se fueron dando

con la evolución de la interfaz.

- El colectivo ACORDE adoptó el desarrollo con PureData, MobMubPlat, Processing y el desarrollo de interfaces gráficas de usuario (GUI), con el objetivo de desarrollar un pieza musical en ensamble con dispositivos móviles.

  El colectivo ACORDE inicialmente usaba PureData como una de sus herramientas por lo que tenían experiencia en el desarrollo de instrumentos digitales desde esta plataforma. Este lenguaje puede ser extendido con la plataforma para Android MobMubPlat que permite la ejecución de patches de PD en un emulador que incluye interfaz gráfica. De la misma forma se usó el lenguaje Processing para desarrollar la GUI huésped y un servidor principal para la generación de audio.

  El grupo requería desarrollar:

  - Un instrumento digital de la misma forma en que lo venían haciendo, es decir usando PD, este instrumento debía ser ejecutado en tiempo real y en ensamble.
  - Una interfaz gráfica para ejecutar el instrumento en celulares Android usando MobMubPlat y Processing, esta interfaz debía ser controlador para un único motor de audio.
  - Un servidor de audio con capacidad para ejecutar la pieza en vivo. Este servidor era en encargado de producir audio para todos los controladores.
  - Una interfaz gráfica para el servidor, donde pudiera ser configurado por cualquiera de los integrantes del ensamble de manera rápida.
- Cinevivo, por su constitución interna y gran cantidad de roles de ejecución e interacción en vivo, además por el gran numero de piezas a construir, adopto tanto GUI, como interfaces física y por último la interfaz textual con técnicas del *live coding*, para el desarrollo del contenido audiovisual.

  El trabajo realizado por el colectivo estaba basado en el uso de software comercial controlado por interfaces MIDI comerciales, por lo que el paso natural era mantener la dinámica original para empezar y evolucionar desde este punto de la siguiente forma.

  - Desarrollar un software con las prestaciones a la que estaban acostumbrados, pero con la posibilidad de ser extendido en etapas posteriores, todo con la filosofía *open source*, en repositorios públicos y donde cada integrante tuviera acceso constante al código fuente.
  - Mantener en primera instancia la interfaz MIDI, pero desarrollar aproximaciones diferentes cada vez. El proceso debía llevar al desarrollo de al menos una interfaz física construida en su totalidad de forma personalizada.
  - La introducción del *live coding* como técnica de interfaz para tener un control más amplio sobre la pieza pero usando todos los comandos e instrucciones ya depuradas a lo largo del proceso.
- El colectivo Algo0ritmos, por su cercanía con el lenguaje chucK, adoptó Chmusick, con el objetivo de participar en un workshop especifico de programación en tiempo real y sumarse también con una participación masiva en un Algorave. Dado que el colectivo contaba con personas en nivel diversos de programación la librería contaba con un estado de trabajo estándar que podía ser extendido para los más experimentados, pero al mismo tiempo presentaba un marco de trabajo estable y simple para los novatos.
- La Cybernetic Orchestra adoptó CQenze para incorporarlo en una presentación internacional durante la *Ïnternational Conference on Live Coding"* como una de las herramientas que cada uno de sus integrantes podía elegir para su participación, añadiéndolo a su motor Estuary. El lenguaje como tal ya está diseñado, por lo que restaba:

- Implementación del *parser* dentro del servidor en linea, para ser usado en tiempo real por cualquiera de los miembros de la orquesta, usando un interprete con un alto nivel de sensibilidad donde no se requerían comandos extras para la obtención de respuesta.
- Sincronización en la interpretación de múltiples lenguajes dentro del servidor, de esta forma los nuevos lenguajes integrados en la plataforma podían ser sincronizados para funcionar en el mismo *performance* de manera colectiva.

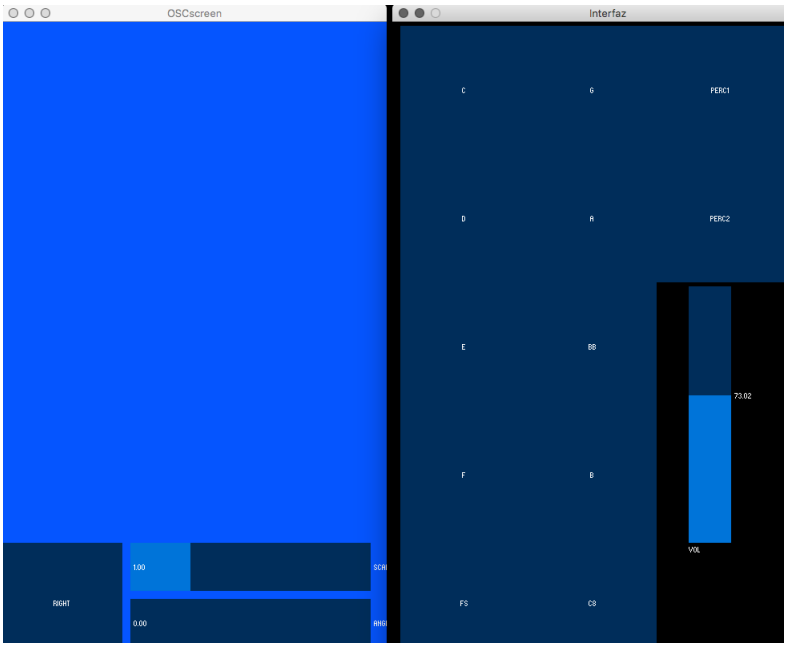
5.1.2. Fase 2: Tutorial y Proceso

Para esta fase, en algunos casos se le entrego la herramienta a cada grupo con el fin de que se familiarizaran, y en otros casos se desarrollo de la mano de cada uno de los usuarios y colectivos. Durante esta etapa, cada grupo tenia la oportunidad de pasar un tiempo con el desarrollador y autor de esta tesis para aclarar dudas sobre el uso. El tiempo fue variable con cada grupo, dado que la experiencia previa en el uso de herramientas similares o la curva de aprendizaje de una a otra variaba.

Así mismo, en todo momento se tenía acceso a la documentación, código fuente y prototipos para revisar posibles mejoras o adaptaciones antes de ser usadas en vivo. A continuación se detalla el trabajo realizado con cada colectivo.

- El colectivo ACORDE tuvo un tiempo presencial de capacitación de tres (3) horas, divididas en dos sesiones. Además, el grupo tuvo acceso completo al código fuente, tanto del servidor, del motor de audio y de la GUI desarrollada para dispositivos Android. Para esta revisión y autoaprendizaje el grupo tuvo una semana.

Figura 5-1.: Interfaz Gráfica para Android



La interfaz que puede verse en la figura **5-1**, fue escrita durante una sesión de reunión junto al semillero usando Processing y MobMubPlat. En esta misma sesión se implemento un prototipo para el

servidor de audio.

En el segundo encuentro se hizo una depuración de errores encontrados por cada usuario.

- El colectivo Cinevivo realizó un proceso distinto, según la herramienta usada y la pieza desarrollada.
  - Para la pieza audiovisual RUIDO (primera ejecución) se eligió el trabajo con una interfaz que hiciera puente entre el trabajo previo y el que se desarrollaría con esta investigación. Por está razón, se mantuvo el uso de una interfaz física comercial, pero en este caso con un software ajustable y personalizado. El tiempo de capacitación presencial fue de una (1) hora y una semana de trabajo individual para la familiarización y práctica.
  - Para la pieza NIHIL CINEMA la interfaz comercial se sustituyó por una interfaz construida específicamente para cumplir los requerimientos de esta nueva experiencia artística. Por las características de la interfaz diseñada no hubo tutorial presencial, sólo una semana de trabajo individual y exploración.
  - La tercera y cuarta prueba se realizaron sobre las piezas ya ejecutadas, haciendo cambios a las interfaces para probar diferentes enfoques y atendiendo a detalles observados durante los *performances*. A RUIDO se le añadió una GUI para acompañar el trabajo de la interfaz física y a NIHIL CINEMA se le cambio el dispositivo físico por uno más ergonómico, sin cambiar el funcionamiento inicial.
  - Random Corpus Binary fue la última pieza, para este caso se movió todo el sistema a una interfaz textual, con un mini lenguaje de programación embebido en C++. El tiempo de capacitación presencial fue de tres (3) horas y una semana de trabajo individual.
- El colectivo Algo0ritmos participó activamente del proceso de desarrollo de la librería Chmusick, por lo tanto no hubo necesidad de un tiempo de capacitación presencial, tuvieron acceso al código fuente y documentación. EL tiempo de familiarización fue de tres (3) días a modo de ensayos de ensamble musical.
- La Cybernetic Orchestra tuvo dos (2) sesiones de dos horas, para usar la herramienta sin ningún tipo de instrucción previa pero con el desarrollador presente. El interprete estaba disponible online para ensayos particulares durante quince (15) días a partir del primer ensayo.

5.1.3. Fase 3: Actos en Vivo

Dado que la naturaleza de las piezas artísticas desarrolladas en el marco de esta investigación es efímera y performática, y teniendo en cuenta la participación del error como un actor más dentro del trabajo colectivo, se le pidió a cada uno de los grupos que ejecutarán en vivo sus actos usando las nuevas herramientas que se tenían a disposición.

La ejecución debía ser en público, en un lugar apto y con las condiciones adecuadas para reducir al máximo otros factores que pudieran inducir cambios o afectar la ejecución de la obra misma. A continuación se narra de forma corta el proceso de interacción de los artistas con sus herramientas, junto a una corta descripción de las piezas desarrolladas desde un aspecto técnico y los problemas no asociados al trabajo específico.

- ACORDE eligió para su presentación la sala de conciertos de el centro cultural La Pascasia, en la ciudad de Medellín.

Figura 5-2.: Ensayo previo al concierto del colectivo ACORDE

Se puso a su disposición un sistema de sonido completo y configurado, operado por el ingeniero residente. Así mismo, un servidor de red configurado y dispuesto para el ensamble. Como puede verse en la figura **5-2** cada integrante del ensamble hacia uso de un teléfono celular, con la interfaz gráfica configurada para acceder el servidor y producir audio.

La pieza elegida para ser interpretada fue - In C - de Terry Riley, en una configuración donde cada artista estaba dispuesto en un punto dentro de la audiencia conectado de forma inalámbrica al sistema general, cada interprete tenía su interfaz GUI ya configurada en su propio Andriod.

Esta pieza requería que cada interprete estuviera en constante uso de la interfaz durante toda la pieza.

- Los *performances* de Cinevivo se describirán independientemente, así:

  - La primera versión de RUIDO se realizó en un teatro amplio de la biblioteca de Belén, en la ciudad de Medellín. En la figura **5-3** puede verse el afiche publicitario de esta pieza.

    El sistema constaba de dos (2) proyectores digitales, sistema de audio completo operado por el ingeniero de sala, y un equipo de proyectores obsoletos, artesanales, entre otros.

    La operación del sistema requería de tres (3) artistas sincronizados y realizaba fragmentaciones y video mapping en tiempo real.
  - NIHIL CINEMA se probó en su primera vez, en el Teatro municipal Guillermo Valencia de la ciudad de Popayán, con un sistema inalámbrico inestable, emitido desde un teléfono celular Android de gama baja, al que debían conectarse dos (2) interfaces para controlar el software a distancia. Es decir, la red se generó haciendo un punto de red desde un dispositivo *smart phone*. Además, un sistema de sonido completo sin operario en sala.

    El afiche publicitario para esta pieza puede verse en la figura **5-4**.
  - RUIDO, como segunda ejecución se realizó en el teatro del Museo de Arte Moderno de Medellín (MAMM). Para esta oportunidad se dividió el trabajo a dos (2) operarios por estación, asi,

Figura 5-3.: Afiche para la pieza RUIDO, pieza trabajo del semillero Cinevivo

ingeniero en sala para el sonido, técnico de luces y escenario.

Esta misma versión se repitió en el auditorio Fundadores de la ciudad de Manizales en la clausura del ISEA'17.

- Para la segunda versión de NIHIL CINEMA se eliminó la red inalámbrica, realizando la conexión por cables.

- Randon Corpus Binary es una pieza para 7 operarios, cada uno trabajando independientemente, encargado de su proyector o sistema de sonido. Esta pieza se ejecutó en vivo en el ICLC realizado en el auditorio del CMMAS de la ciudad de Morelia en México. La interfaz es independiente y no requiere de ajustes de red.

La figura **5-6** muestra la interfaz física que se uso durante la presentación de la pieza para modificar parámetros de el motor de machine learning, que se usa para la ejecución en vivo de esta pieza.

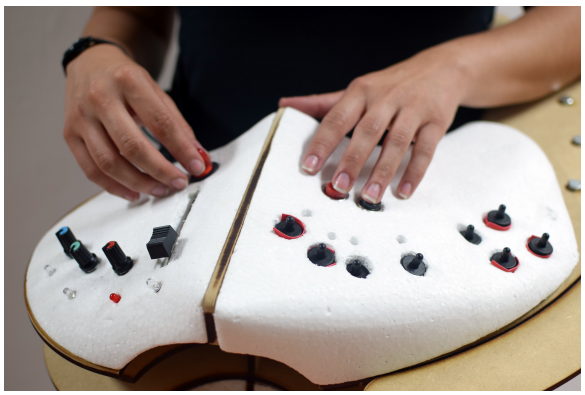
- El sistema para la pieza de la Cybernetic Orchestra es un servidor robusto en un sistema de red al que se accede por cable, y al que se añaden otros artistas desde internet. La pieza se ejecutó en vivo también en el ICLC realizado en el Algorave de clausura de la ciudad de Morelia en México.

  La figura **5-7** muestra la pantalla y la disposición de los ejecutantes presenciales durante el performance de la Cybernetic Orchestra en Morelia. El sonido estaba manipulado por ingeniero en sala.

- Algo0ritmos escogió el centro cultural de Platohedro, el Museo Universitario de Ciencias de La Salle y un streaming mundial a través de youtube. En cada caso el sistema de audio, de streaming y red estuvieron operados por técnicos capacitados. Se decidió hacer una pieza desde cero, improvisando el código en un entorno colaborativo.

Figura 5-4.: Afiche de la pieza Nihil Cinema, pieza trabajo del semillero Cinevivo

Figura 5-5.: Interfaz Física (MIDI - OSC) para controlar Software en vivo



Este acto en vivo constaba de programadores que podían cambiar los algoritmos de forma colectiva y a lo largo del performance por turnos en la salida de audio.

5.1.4. Evaluación

Observación

Se escogió la observación directa por parte del desarrollador junto a la del director del ensamble particular, como un método que podía dar luces en la evaluación, dado que el proceso es altamente subjetivo, considerando también no se esperan resultados precisos y medibles, sino más bien intuiciones de cómo se abordó el proceso de creación artístico en el contexto de los nuevos medios, entendiendo este proceso como la relación entre el artista y su interfaz.

De la misma forma, la observación de los cambios de la pieza puede no ser vista desde la perspectiva del público observador o de los mismos ejecutantes, que por el estrés del acto en vivo pueden no ser conscientes

Figura 5-6.: Interfaz Fisica (MIDI - OSC) para controlar Software en vivo

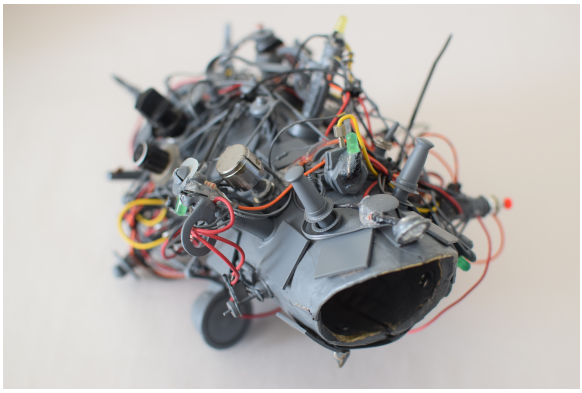


Figura 5-7.: Cybernetic Orchestra en ICLC'17 Morelia - Méx. 8 participantes en escenario, más 12 personas remotamente.

de la obra durante su ejecución. Así mismo, el contacto directo con los artistas para ir descubriendo sus incomodidades en el proceso, empezando desde los prototipos hasta la versión funcional, sólo podía darse de esta forma, por lo que el proceso mismo de desarrollo y los cambios de la interfaz pueden interpretarse como variables a evaluar, las cuales son el resultado de un proceso conjunto entre la pieza, el artista, el desarrollador y el publico.

Opiniones de los Artistas

Además de la observación, al terminar cada ejecución, se hizo una evaluación colectiva, en la que el desarrollador hacía parte sólo de manera pasiva. En esta evaluación se valoraba la pieza artística como un todo, y al mismo tiempo se hacía un recapitulación por partes, a modo de retroalimentación.

Cada uno de los actores en escena podía dar sus opiniones de forma espontánea. Cuando las opiniones tenían algo importante a ser rescatado para los propósitos de esta investigación, se procedió a pedir ampliaciones

para descartar los factores externos, cuando fuera posible.

De estas opiniones surgieron los cambios posteriores y las nuevas ideas, es decir, cada nueva propuesta en términos de interfaz o de modelo general surgió como resultado de estas conversaciones por lo que la evolución de las interfaces puede verse como el resultado concreto. Estás opiniones fueron grabadas en audio para su posterior evaluación.

5.2. Hallazgos

De forma breve se describirán algunos de los hallazgos evidenciados durante los actos en vivo descritos previamente.

- La experiencia del semillero ACORDE es clara para extraer un resultado que esta ligado a la apropiación y empoderamiento de las herramientas, cuando estas son desarrolladas por los artistas de forma activa, o cuando durante el proceso de construcción de estas herramientas los artistas están como entes activos en su construcción y evolución. Cada ensayo y acto en vivo se transformó en un laboratorio de experimentación donde la modificación y mejoramiento de las interfaces era el punto principal de discusión. Las opiniones generales empezaron a fluir de forma natural, transformando los actos en vivo en un laboratorio vivo de desarrollo en tiempo real de interfaces.
  La explotación de los errores en las herramientas fue también un punto fuerte al ver este como parte del proceso constructivo-deconstructivo y generando más interés en el proceso de mutación de la interfaz misma que en la obra como resultado del proceso.
- El trabajo con el semillero Cinevivo abrió diferentes perspectivas a personas que no tenían experiencia previa con lo que hay tras bambalinas en el proceso de desarrollo de las interfaces que se usan durante la ejecución en vivo de sus piezas audiovisuales. Gracias a esto, los integrantes se interesaron en los detalles que de otra forma hubieran permanecido ocultos en el código fuente de un software comercial, ayudándoles a comprender mejor su propio quehacer y el proceso mismo de conceptualización hasta la materialización y ejecución en vivo.
- Tras el proceso con el colectivo Algo0ritmos, uno de los hallazgos más importantes fue la necesidad de mantener una documentación constante de los cambios implementados en las interfaces, ya que cada cambio lleva a resultados inesperados, y es de vital importancia poder tener un registro de los cambios y el como resultó la obra tras la implementación de estos cambios.

  Una consecuencia directa de esto es la posibilidad de escribir la experiencia de manera formal, para que esta pueda ser usada como guía de trabajo, como ejemplo, o como muestra de errores que pueden ser evitados o buscados, logrando además una visibilización y la generación de una comunidad de discusión y aprendizaje.

  Chmusick se ha convertido gracias a este esfuerzo en una herramienta para la experimentación, pues al poder construir versiones diferentes de prueba, con aproximaciones sintácticas diferentes, ha originado una rápida evolución de la herramienta hacia caminos insospechados.
- En el proceso de implementación de CQenze sobre Estuary, la Cybernetic Orchestra aportó una observación para el trabajo que confirma las bondades de este como un lenguaje de primera experiencia, ya que brinda una experiencia rápida y significativa para lograr motivar a los curiosos en el *live coding*. Pero más que esto, durante la implementación se pudo observar todo el tiempo el interés que genera desarrollar nuevas sintaxis, o nuevas relaciones semánticas sobre sintaxis existentes, y como

este proceso puede enriquecer de manera abundante el quehacer artístico y expresivo en ensambles y grupos colaborativos, donde el espacio se presta más para la diversión y la comunicación que el virtuosismo, en el uso de lenguajes más complejos.

5.3. Discusión

Luego de participar en cuatro procesos creativos, con necesidades diversas y resultados artísticos no equiparables, es importante notar que uno de los resultados que si se puede resaltar de forma general, es que tanto el director artístico de los colectivos, como los artistas, perciben un incremento en el grado de compromiso para entender la interfaz, familiarizarse y lograr un grado aceptable de experiencia, dado que conciben la interfaz como suya, y esta concepción agrega también un nivel de esfuerzo y autorregulación, que no es tan notorio con el uso de interfaces desarrolladas en procesos que no incluyen al usuario final como un elemento activo. Esto es consecuencia de que los artistas fueron entes activos en todas las fases de desarrollo de las herramientas, que luego ellos mismos van a usar para sus actos en vivo.
De lo anterior se pueden extraer dos resultados que son consecuencia directa. El primero, es que la frustración que puede presentarse por no entender el uso de un interfaz se transforma en un deseo de hacer mejoras y mutaciones a la misma. El segundo, es la cada vez más efectiva capacidad de comunicación entre artista - interfaz - obra - y estas, luego de ser evaluadas critica, pero subjetivamente, y se transforman en nuevas ideas y procesos de mejora o transformación, que pasan luego al desarrollador para iniciar de nuevo el ciclo.

Otra consideración que puede hacerse, es que sin importar la obra resultante los usuarios estaban cada vez más interesados en aplicar cambios, para intentar nuevos resultados en vivo. Por consecuencia, la aceptación al error se transformó en un pilar, tanto en el desarrollo como en la ejecución, llevando el proceso a lugares insospechados e impensables en ambientes donde se esperarían resultados depurados y estables.
En este punto se puede ir más lejos, dado que al artista empoderado en su trabajo, y entendiendo su nuevo papel como constructor - artesano de sus propias herramientas, también ve la interfaz (que el mismo construyó o ayudó a construir) como parte integral de la obra, marcando una diferencia entre los nuevos medios con otras formas artísticas, dándole estatus de obra también a la interfaz (proceso) misma.

Finalmente es importante decir que es un trabajo que requiere mucho tiempo, dedicación y un grupo de trabajo amplio (a falta de una mejor palabra para enmarcar estos procesos colaborativos) dado que por la naturaleza misma de los procesos, este tipo de trabajo seria impensable sin estas condiciones mencionadas.

5.4. Conclusión

Durante dos años se trabajó en tres procesos diferentes de desarrollo que incluyeron a cuatro colectivos y/o semilleros, observando y experimentando, en un proceso que se dejo fluir espontáneamente y que llevo a resignificar y recategorizar, al menos en el contexto de esta investigación, y en los procesos artísticos de los colectivos participantes la forma de ver al artista, tranformandolo por su participación en el desarrollo de las herramientas en un luthier, elevando el proceso de la obra (la concepción, conceptualización y desarrollo) al nivel de la obra misma, poniendo la interfaz al nivel de obra (paralelamente a la obra producida con esta) e involucrando a los desarrolladores haciéndolos parte activa de los colectivos en su proceso creativo no sólo dando soluciones técnicas a problemas de diseño o implementación sino aportando con otras perspectivas sumando así a los resultados artísticos.

Historia IV - Caos y Cosmos

¿Es el ser humano un registro estático, ya definido y terminado? No, no es esta la pregunta, la pregunta es ¿cuál es tu canción favorita de los Bee Gees? Y realmente la primera y la segunda pregunta confluyen en el mismo desarrollo temático, no importa si alguno tiene la razón, seres construidos de lenguaje simbólico incapaces de acceder a algo distinto que a sus propias memorias codificadas a lo largo miles de años de humanidad, y a las memorias que otros seres han dejado aquí o allá usando la técnica como mensaje al futuro, luchando por tratar de entender eso de ser y no ser, o de ser como si uno realmente fuera algo más. Están ahí, conduciendo un automóvil entre una fuga oscura, un intrincado laberinto de símbolos sin esperanzas de salir, y ahí está: no eres tú pero "how deep is your love". Diez años y ¿crees que nunca he escuchado a los Bee Gees?, no, nunca lo has hecho, yo te conozco, no eres tu. El motivo inicial se oculta un rato, y durante este contrapunto el silencio de la noche con el susurro de la lluvia, marcarán el final de la lucha entre el soy o seré, o tal vez fuí.

6. Conclusiones y Perspectivas

La presente investigación se realizó en un contexto completamente práctico y en un proceso enmarcado en la prueba y el error. De esta forma, las conclusiones conducen a nuevas preguntas, o al menos, como semillas de nuevas preguntas que además pueden servir para pensar sobre el futuro de los procesos creativos en las artes vivas, sobretodo en grupos colaborativos y colectivos artísticos diversos.

6.1. El proceso se hace obra

Sobre el cómo la interfaz impone una forma de materializar las ideas artísticas

Cuando se habla de artes vivas de la mano de los nuevos medios, uno de los primeros problemas ante los que un artista (en el caso de esta investigación, colectivos artísticos) se enfrenta, es a la decisión, casi involuntaria, de escoger una interfaz, que le permita establecer un proceso de comunicación transparente entre sus ideas y las herramientas, para traer de la nada (su mente) a la materialidad, su obra (poiesis). Este proceso, que es enfrentado de forma inconsciente, termina por ser determinante, tanto en el proceso creativo como en los resultados, dado que el lenguaje, los métodos y la comunicación en si misma van a estar siempre ligados al tipo de interfaz escogido[4]. Es que la interfaz como objeto limitado y limitante no puede ser diseñada con posibilidades infinitas.

La mente del diseñador queda plasmada en su obra (la interfaz) y ésta plasma con su propia firma la obra que esta interfaz facilite o ayude a materializar. El proceso creativo no es un hecho aislado y desconectada, y al contrario es un engranaje y tejido al mismo tiempo, que al ser analizado no ya holísticamente sino ecológicamente [18] muestra tantas variables como posibilidades; en esta linea de pensamiento, podemos añadir, tantos errores como limitantes.

Es claro también, que el medio escogido conlleva una decisión tanto metodológica como práctica (y en muchos aspectos, también política, más si se habla del software [15]), que influye de manera natural tanto en las narrativas como en el proceso práctico de ejecución.

Ahora, si la interfaz, que, como ya se dijo juega un papel vital en el desarrollo de procesos artísticos en ambientes tanto electrónicos como digitales, está cargada (como toda herramienta) de una fuerte memoria [28], estos procesos son tornados violentamente hacia cierta forma específica de hacer las cosas. Violentamente porque es impositiva, así esta imposición pase desapercibida para el usuario y porque en este artefacto que es la interfaz se solucionan unos problemas; problemas que el desarrollador considera importantes y a los cuales quiere enfrentar con posibilidades en su diseño, diseño que al mismo tiempo, debe dejar por fuera la resolución de otros problemas no previstos o ante los cuales aun no se vislumbran soluciones.

Ante esta perspectiva, el artista está tentado a aceptar un lenguaje estandarizado y permitir que su idea se transforme a lo que puede o no puede comunicar su interfaz, o puede embarcarse en la construcción de sus propias interfaces (su propio lenguaje, su voz), asumiendo este nuevo proceso, no contemplado inicialmente como una obra en sí misma (poiesis de nuevo), trayendo de su imaginación tanto la obra-resultado como las

herramientas-interfaces a la materialización. No necesariamente con sus propias manos, y no con el animo de volver productivo y comercial su desarrollo (tampoco excluyendo esto) pero si enfocado en la decolonización [27] del si mismo y en el planteamiento de las preguntas ontológicas en la creación artística.

En este proceso debe estar dispuesto a aceptar el error como natural, no solo para sí mismo, sino como una cualidad inherente al proceso artístico, que ahora debe mutar paso a paso. Este error, se presentará como una forma de disrupción ante la imposición, y este método posibilitará el hackeo mental tanto del artista como del artesano y al mismo tiempo del público testigo. Una de las consecuencias de esto es que su trabajo deja de ser solitario, ya que debe incluir nuevas perspectivas, habilidades y conocimientos en su creación-ensamble y al mismo tiempo el público se añadirá de forma activa a la experiencia evolutiva de cada obra.

Así mismo, es posible entonces afirmar que este proceso, lejos de enmarcarse en un proyecto de diseño lineal, se define a sí mismo como un proceso espiral, donde ninguno de los actores está buscando un producto terminado, sino más bien, entiende que el proceso de construcción (o deconstrucción, cualquiera sea el caso) es un proceso constante, con resultados nuevos a cada paso.

Es aquí donde la violencia entra en la ecuación, para trastornar la forma de hacer, permitiendo reiniciar los ciclos creativos con cada nueva ejecución en vivo y de esta forma se plantea una política del error como método decolonizador donde el arte en este caso, juega a ser libertino, permitiendose evolucionar y revolucionarse con cada nueva ejecución.

- El usuario liberado de la tiranía e impotencia ante la interfaz toma riesgos a pesar del resultado-obra, elevando el nivel del proceso de construcción a la obra misma, siendo violento tanto consigo mismo, como con sus ideas, procedimientos y pre-conceptos.

El diseño de interfaces para la música ha recorrido un camino exitoso en el área de los nuevos medios, convirtiéndose en un campo de estudio específico con características propias (NIME) y es posible aventurarse a pensar que se expandirá, tanto a otras artes (ya lo viene haciendo) como a contextos diferentes, donde nuevas formas (formas propias, formas autóctonas, formas regionales, etc.) de enfocar los procesos de comunicación, puedan plantear soluciones creativas (o algo más importante, nuevas preguntas) y sobretodo, aportar en la búsqueda de nuevas formas de resistencia-arte ante los sistemas hegemónicos, y al nuevo paradigma de la contemplación artística como producto de consumo.

6.2. Pequeños cambios - Nuevos lenguajes

Sobre como el live coding ejemplifica la adopción de los cambios en su evolución natural y técnico

Como ya se ha mencionado varias veces, embarcarse en el camino de desarrollar nuevas herramientas-interfaces, en este caso para la materialización de procesos artísticos, requiere de tiempo, esfuerzos técnicos, presupuesto y renuncia (el proceso artístico/artesanal), pero sobre todo de un equipo de trabajo colectivo, con habilidades y puntos de vista distintos que permitan llevar los desarrollos a nuevos enfoques supliendo o facilitando las limitantes arriba mencionadas. Aunque es posible introducir también alteraciones más pequeñas, pequeñas variaciones que también logran dar una identidad única a los procesos creativos y donde la correspondencia entre el gesto y el resultado artístico no está atado sólo a las decisiones de diseño, enmarcadas en el desarrollo original (y por lo tanto comercial) de la interfaz, sino también en la experiencia y capacidad expresiva del artista.

En este contexto, la aparición de microcontroladores fáciles de programar, como Arduino (y toda la variedad de nuevas marcas con prestaciones y agregados diferentes), y la popularización de impresoras 3D (también

los nuevos materiales y herramientas), han facilitado enormemente las posibilidades para el desarrollo de nuevas formas de trabajo, y al mismo tiempo han abierto el campo del desarrollo de interfaces (antes sólo para grandes corporaciones y laboratorios de investigación), haciéndolo económico y accesible al público general, dándole la capacidad de experimentar con interfaces físicas diversas.

Por otro lado, el *live coding* se aprovecha de su amplia definición, al mismo tiempo de su mutabilidad, para ser un campo de exploración artístico donde la creatividad se expresa en el desarrollo de lenguajes (herramientas, enfoques, giros, tanto sintácticos como semánticos) para expresar ideas de diferentes formas. Es en este nuevo camino de exploración, donde la mayor cantidad de nuevos lenguajes de programación se han desarrollado en la última década, y se ha dado pie al concepto de DSL, en los que no es necesario contar con lenguajes complejos y monumentales, llenos de prestaciones y con entornos de desarrollo muy pesados. La anterior es una consecuencia natural de que las tareas a realizar pueden ser completadas con desarrollos más ligeros anclados a la gran capacidad de cómputo, y el acceso a nuevas formas de desarrollar tanto analizadores léxicos, como funciones de mapeo rápido entre los lenguajes desarrollados y la interpretación en tiempo real.

Así mismo, la implementación de protocolos de red (tanto internos como externos) como Open Sound Control (OSC), permiten la modularidad y cooperación entre diferentes lenguajes y aplicaciones, proporcionando a cada artista-desarrollador la capacidad para implementar interfaces únicas, que pueden adaptarse fácilmente en el ecosistema general y en el flujo de trabajo de otros artistas, facilitando de esta forma la conformación de equipos de trabajo diversos, ensambles y orquestas, potenciando el trabajo en red.

Todos estos giros han sido ya resaltados en esta tesis en varios momentos, y han elevado, cada uno en su momento y dentro de su ecosistema, un grito revolucionario ante dificultades encontradas tanto por diseñadores/artesanos como por artistas y creativos.

- Cada cambio, sin importar su tamaño aparente, es el reflejo de un individuo buscando su identidad. El acto poiético del individuo trayéndose a sí mismo hacia su materialización como ser único.

Con esto en mente, es posible aventurarse a afirmar que los entornos colaborativos, con protocolos de comunicación definidos o herramientas para el desarrollo a varias manos de forma simultánea (editores de texto múltiple, compiladores e intérpretes políglotas, motores e intérpretes basados en web), así como las interfaces físicas que interactúen en tiempo real de forma inalámbrica como nodos interconectados, serán el camino futuro de las artes vivas, enmarcadas en los conceptos de colaboración, redes y diversidad.

6.3. Hagámoslo juntos

Sobre como el desarrollo de interfaces (sin importar su naturaleza) permite encontrar nuevas formas en el quehacer artístico

Usando el software como "metáfora de la metáfora" [20], es posible ver el problema creativo como una gran tarea que debe ser dividida en módulos más pequeños, donde la tarea de programar o procesar (cualquiera sea el caso) puede (y en el mejor de los escenarios, debe) recaer en máquinas (personas) diferentes; logrando de esta forma una obra-proceso-ensamble complejo pero al mismo tiempo (dinámico y abierto) mutable (mutante) y vivo. Este proceso se plantea como un camino de errores, pero ya no vistos como una peste a eliminar, sino más bien, como un potencial, como un giro y como la puerta para explorar nuevas ideas.

El arte mismo, ya desmitificado, la obra (que ahora también incluye el proceso) y el artista (que ahora es multitud) deben hacer consciente su naturaleza efímera (característica inherente a las artes vivas) pero

también su potencial de transformación (acá de nuevo el error como mutación), para resignificar el trabajo colectivo en los procesos artísticos; donde el colectivo como ser y como sistema, puede presentarse como un pilar para el desarrollo y construcción de nuevos puentes para las artes digitales.

- Una fuerte relación entre el proceso de desarrollo de la mano del artista y la simpatía hacia la interfaz, que de esta forma se transforma en una herramienta de auto conocimiento llena de violencia inherente.

Dicho esto, el trabajo colectivo trae consigo el verbo hacer, conjugado como un nosotros, y a este nosotros lo une el trabajo manual-artesanal, para dar a luz tanto un proceso como un ensamble, una interfaz y una obra. La interfaz entonces deja de ser un medio y la obra un fin, ahora las dos se funden elevando de esta forma el estatus de la interfaz, dado que ahora es obra en sí misma, por lo que se genera una simpatía hacia el artefacto (arte-facto), logrando de esta forma comprender profundamente la relación que las herramientas han tenido en los procesos humanos, relación que tiende a desenfocarse (difuminarse y hasta desaparecer) cuando la producción masiva e impersonal de artículos de consumo es la norma imperante.

Ante esta perspectiva el arte toma de nuevo el estandarte y se proclama como un camino humanizador, donde las manos constructoras (la materia) y el concepto (las ideas) ya no están en un perverso binario (el bien y el mal), sino son múltiples procesos (múltiples dimensiones) entrelazadas (tejidas) por lazos (hilos) creativos en búsqueda tanto de una identidad individual como colectiva (humana).

Historia V - El libro

"Yo testifico a todo aquel que oye las palabras de la profecía de este libro: Si alguno añadiere a estas cosas, Dios traerá sobre él las plagas que están escritas en este libro. Y si alguno quitare de las palabras del libro de esta profecía, Dios quitará su parte del libro de la vida, y de la santa ciudad y de las cosas que están escritas en este libro". Libro del Apocalipsis ... de nuevo la interfaz (el lenguaje natural) que maldice a su usuario... déjame en paz querida mía ... ahora ¡Soy!.

A. Anexo: Para donde va el live coding; Opiniones Personales de Otros Artistas

A.1. posibles -hacia donde-

Esta sección incluye comentarios recogidos en respuesta a la pregunta, *where is live coding going?* donde se les pidió a exponentes de diferentes tendencias y ramas del live coding que de forma espontánea dieran sus opiniones.

Alexandra Cardenas:

"Va a convertirse en uno de los caminos más frescos para la creación musical. Es una gran opción para el futuro de las artes, liberadas del yugo de la industria".

Shelly Knotts:

"Isn't the point that we make it up as we go along?" "Hybrid systems and integrating into the fringes of pop music"

Thor Magnusson (ixi):

"No one can see the future, but I think live coding will see many developments: new languages, new systems, many of whom are visual and physical as well the use of machine learning in coding new communities using live coding methods in their art more visibility of live coding - superstars? live coding eventually disappearing as it becomes part of normal practice and something that does not need to have a unique name".

Juan A. Romero (Benoît and the Maldelbrots):

"Se va a integrar en otras practicas de ejecución y seguirá desarrollando en escenas de improvisación, ya sea de música electronica para bailar o experimental, simplemente se convertirá en otra herramienta para usar en el momento adecuado".

Jaime Alonso Lobato:

"Códigos fuera de computadoras electrónicas. El código como método de enunciación humana haciendo una computadora a base de bacterias y desarrollando un código de programación geométrico (simétrico/asimétri-

co) así también con computadoras químicas y mecánicas como el khipu inca".

Emilio Ocelotl (RGGTRN):

"Va hacia una cuestión performática, la voluntad del Livecoder expresada de una manera compleja, técnica, conceptual y musicalmente. Como una forma de extender las posibilidades de la música por computadora de cara a una especie de acercamiento a la improvisación."

Samuel Aaron (Sonic PI):

"Live coding is going to be obvious."

Malitzin Cortes (cndsd):

"Va para el futuro y la igualdad".

David Ogborn

"Connecting live coding to things that are not live coding"

B. Anexo: Algunos Prototipos, ejemplos de código y fotos de los procesos

B.1. Chmusick

Figura B-1.: Esta es la clase de la cual todos los objetos Chmusick deben heredar

```
public class Chmusick extends Chubgraph
//every class that uses CHmUsiCK members must extend this class
{
    Gain Master => Dyno Processor => outlet;

    120 => static float TEMPO; // Change this to the BPM you want to CHmUsiCK!
    convert(TEMPO) => static dur DTEMPO;

    4 => int OverallDivision;

    4 => static int CYCLES;
    4 => static int MEASURE;
```

Figura B-2.: Ejemplo de una pieza que usa multithreading en la sintaxis simplificada de Chmusick

```
TheCooker  s => dac;
Drum drum => dac;
Harmony h => dac;
FMSynth fm => Echo echo => NRev rev => dac;
Rec r;

[60,62,63,62,70] @=> int notes[];
(32,1) => fm.ratio;

0.5 => rev.mix;
0.6 => echo.mix;

170 => Chmusick.tempo;

spork~drum.drum(drum.favorite(5));
spork~h.sinOsc(["Cm","Cm","Ab","Ab"]);
//spork~h.sinOsc(["Bb"]);
spork~fm.fmBass([48,0,48,0,0,0,0,0,0,0,0,0,0,0,0,0]);
spork~s.sound(notes);

day => now;
```

B.2. CQenze

Figura B-3.: Sintaxis de CQenze sobre Estuary

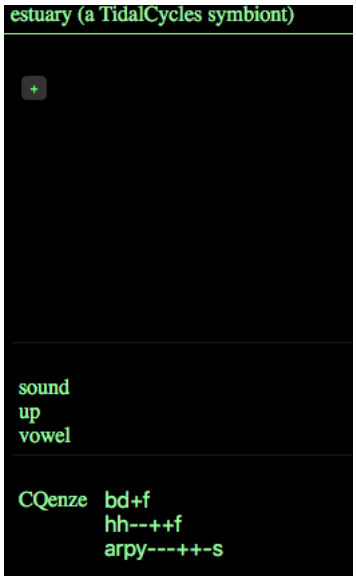


Figura B-4.: Patrón de color producido en CQenze, interpretado en Javascript

B.3. Cinevivo

Figura B-5.: GUI en Android para mapping en tiempo real

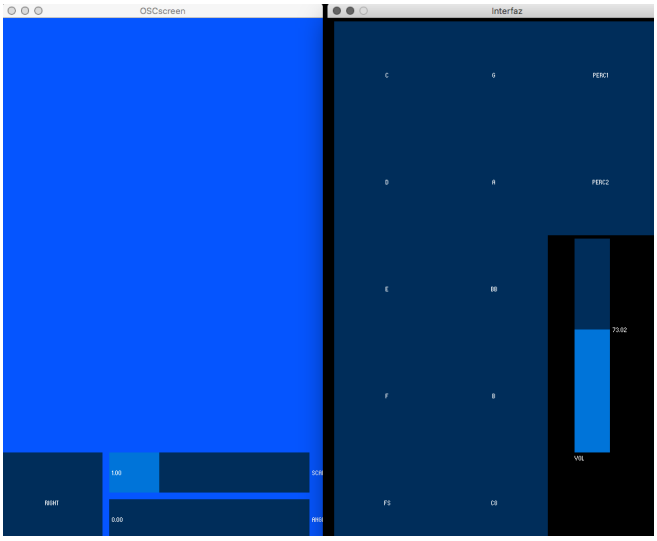


Figura B-6.: Software Cinevivo

Figura B-7.: Ejemplo de sintaxis mini lenguaje Cinevivo

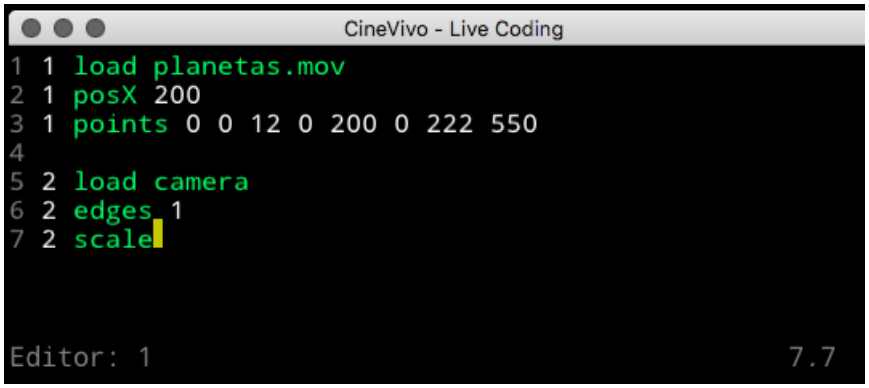


Figura B-8.: Dibujos iniciales, por Luis Rodriguez, fotografía Luis Castro

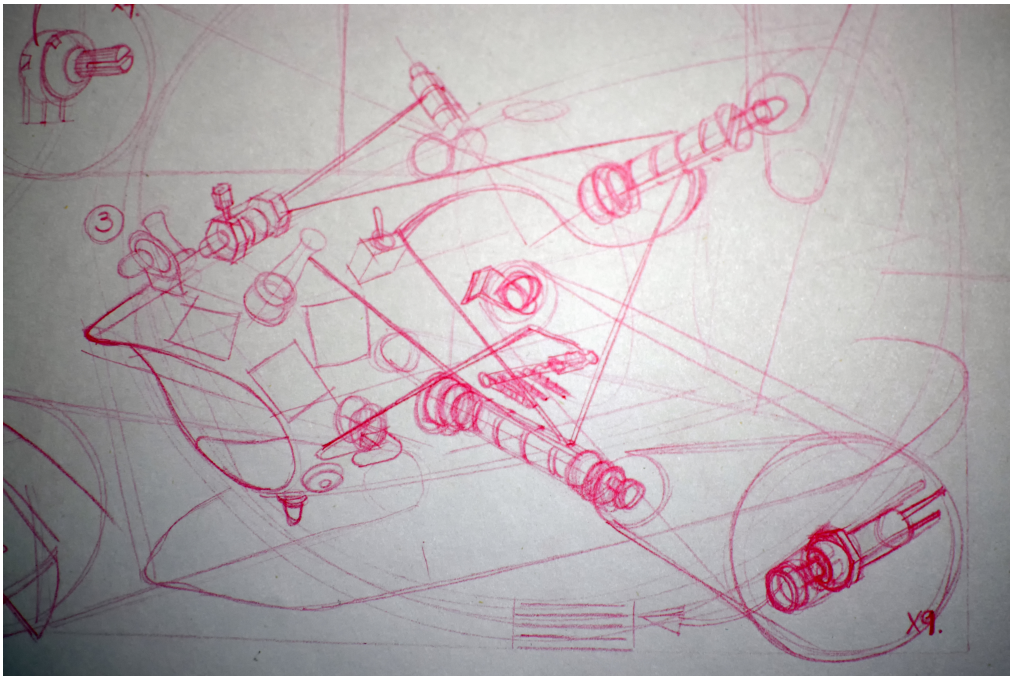


Figura B-9.: Proceso de construcción fotografía Luis Castro

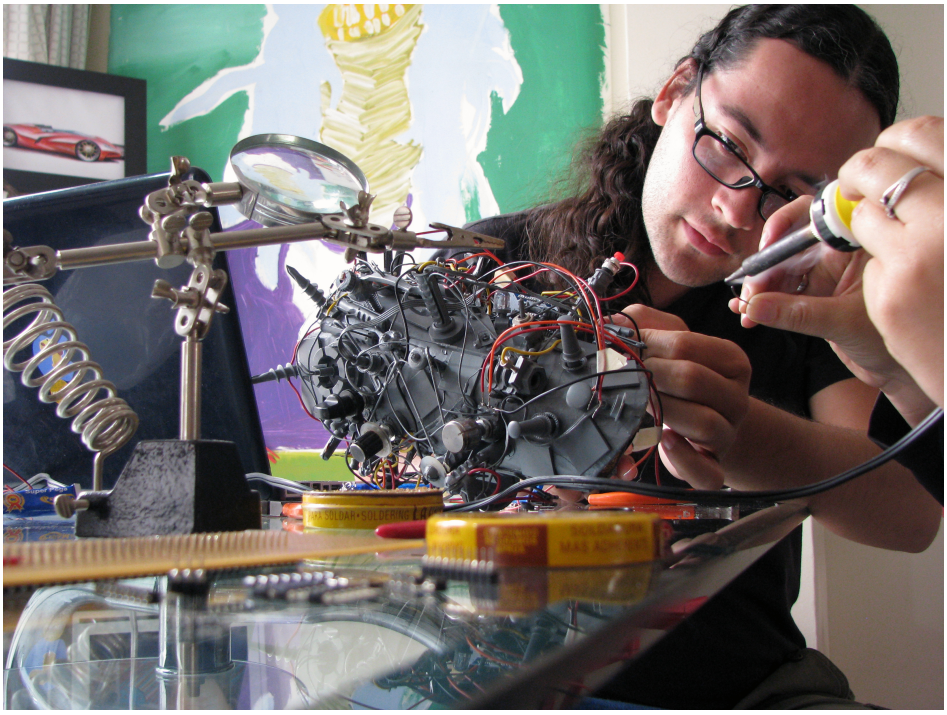


Figura B-10.: Interfaz Completa fotografía Luis Castro

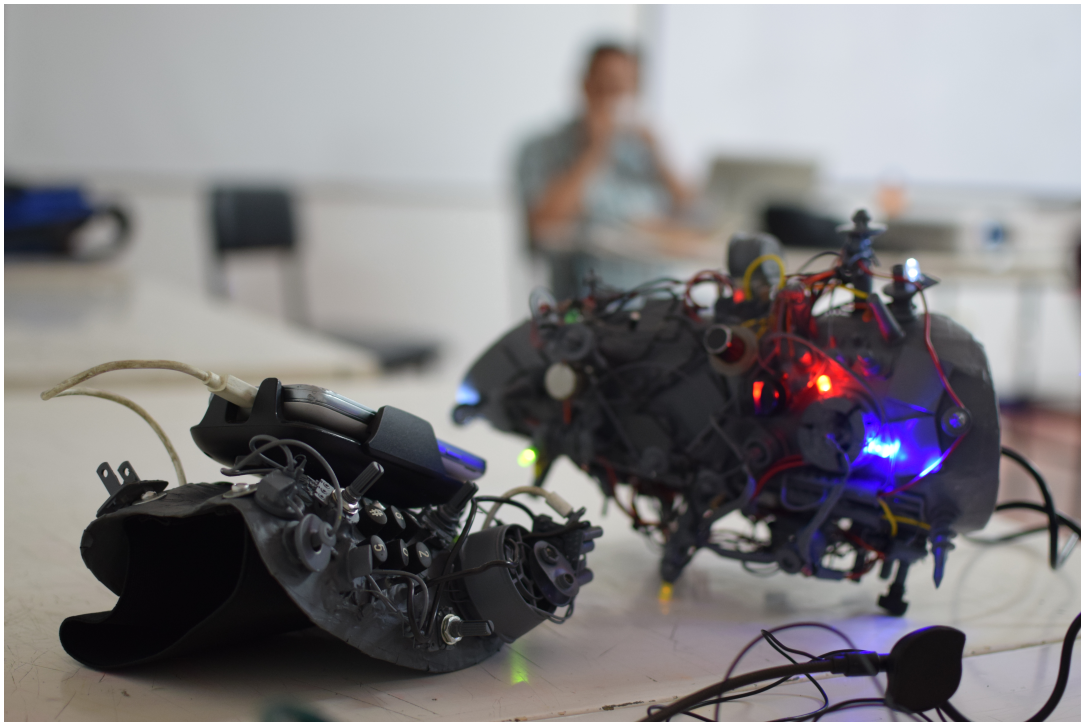
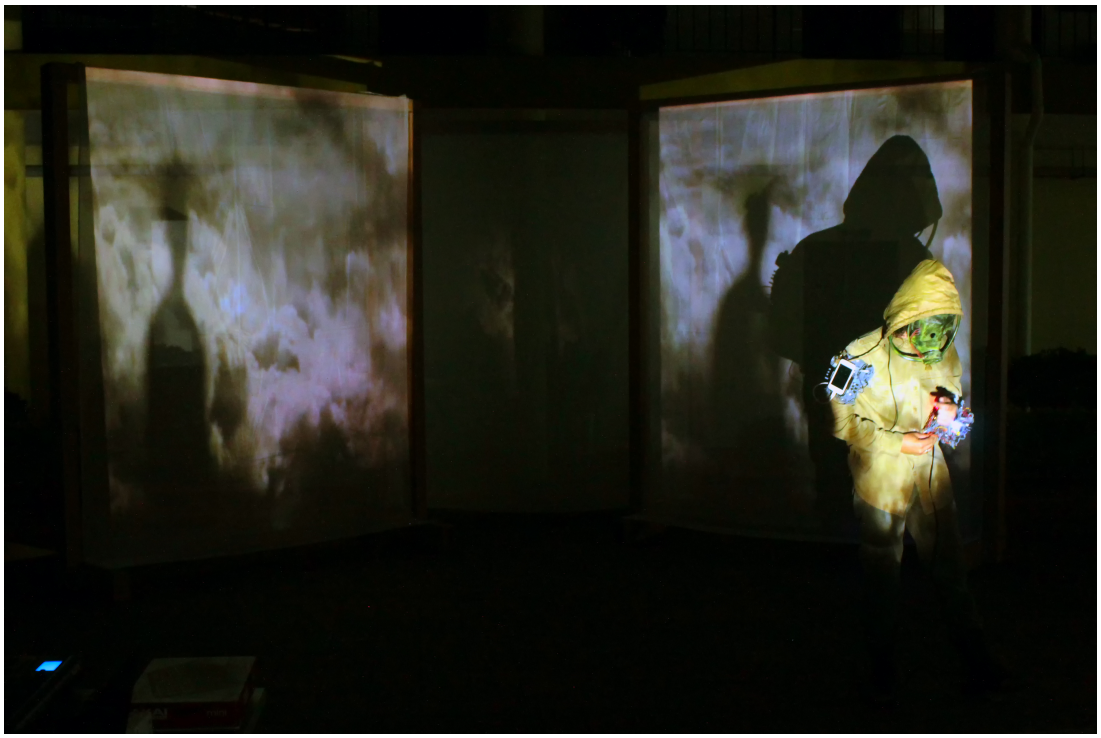


Figura B-11.: Acto en vivo fotografía Luis Castro



C. Anexo: Links a repositorios con el código fuente

C.1. Chmusick

C.1.1. Versión inicial

https://github.com/essteban/CHmUsiCK

C.1.2. Versión con cambios sintácticos

https://github.com/essteban/ChmusickfuncTest

C.2. CQenze

https://github.com/essteban/CQenze

C.3. Cinevivo

C.3.1. Versión inicial

https://github.com/essteban/CineVivo

C.3.2. Versión para live coding

https://github.com/essteban/cinevivoLC

Bibliografía

[1] *All about live coding.* https://github.com/lvm/awesome-livecoding/blob/master/README.md. – Accessed: 2017-11-26

[2] *Generative Music.* http://www.inmotionmagazine.com/eno1.html. – Accessed 8 Diciembre 2017

[3] Volume Information. En: *Natural Language & Linguistic Theory* 14 (1996), Nr. 1. – ISSN 0167806X, 15730859

[4] Código Abierto — Obra Abierta: El Efecto de las Prácticas de Intercambio de Código en la Composición de Música. (2012)

[5] La investigación en artes: el problema de la escritura y el "método". 5 (2012). – ISSN 1794–6670

[6] Aaron, Samuel: Sonic Pi – performance in education, technology and art. En: *International Journal of Performance Arts and Digital Media* 12 (2016), jul, Nr. 2, p. 171–178. – ISSN 1479–4713

[7] Aaron, Samuel ; Blackwell, Alan F.: From Sonic Pi to Overtone: Creative Musical Experiences with Domain-specific and Functional Languages. En: *Proceedings of the First ACM SIGPLAN Workshop on Functional Art, Music, Modeling & Design* (2013), p. 35–46. ISBN 978–1–4503–2386–4

[8] Aaron, Samuel ; Blackwell, Alan F. ; Burnard, Pamela: The development of Sonic Pi and its use in educational partnerships: cocreating pedagogies for learning computer programming. En: *Live Coding in Music Education: Special Issue of* 9 (2016), p. 75–94. – ISSN 17527074

[9] Aaron, Samuel ; Blackwell, Alan F. ; Hoadley, Richard ; Regan, Tim: A principled approach to developing new languages for live coding. En: *Nime* (2011), Nr. June, p. 381–386. – ISSN 22204806

[10] Aaron, Samuel ; Orchard, Dominic ; Blackwell, Alan F.: Temporal semantics for a live coding language. En: *Proceedings of the 2nd ACM SIGPLAN international workshop on Functional art, music, modeling & design - FARM '14* (2014), p. 37–47. ISBN 9781450330398

[11] Bell, Renick: An Interface for Realtime Music Using Interpreted Haskell. En: *Proceedings of LAC 2011*, 2011

[12] Betancur, Esteban: Diseño e implementación de un DSL : CQenze, como lenguaje de primera experiencia para el código en vivo. En: *Proceedings Festival Internacional de la Imagen '16* (2016)

[13] Blackwell, A ; McLean, Alex ; Noble, J. ; Rohrhuber, Julian: Collaboration and learning through live coding. En: *Dagstuhl Reports* 9 (2014), Nr. 2, p. 130–168

[14] Boyle, D: Network Musics: Play, Engagement and the Democratization of Performance. En: *Contemporary Music Review* 28 (2009), Nr. 4

[15] Bratton, Benjamin H.: *The Stack: On Software and Sovereignty.* 1st. The MIT Press, 2016. – ISBN 026202957X, 9780262029575

[16] Brown, Andrew R. ; Sorensen, Andrew: Interacting with generative music through live coding. En: *Contemporary Music Review* 28 (2009), Nr. 1, p. 17–29. – ISBN 0749446080

[17] Bukvic, Ivica I. ; Martin, Thomas ; Standley, Eric ; Matthews, Michael: Introducing L 2 Ork: Linux Laptop Orchestra. En: *Proceedings NIME* (2001)

[18] Capra, F.: *The Web of Life: A New Scientific Understanding of Living Systems*. Anchor Books, 1996 (Anchor books). – ISBN 9780385476751

[19] Carmichael, H: Computers, Children and Classrooms: A Multisite Evaluation of the Creative Use of Microcomputers by Elementary School Children. Final Report. En: *Ontario Dept. of Education* (1985)

[20] Chun, Wendy Hui K.: *Programmed Visions: Software and Memory*. The MIT Press, 2013. – ISBN 978–0262518512

[21] Collins, Nick: Live Coding of Consequence. En: *Leonardo* (2011). – ISSN 0024–094X

[22] Collins, Nick: Live coding and teaching SuperCollider. En: *Journal of Music, Technology and Education* 9 (2016), Nr. 1, p. 5–16. – ISSN 17527066

[23] Collins, Nick ; McLean, Alex: Algorave: A Survey of the History, Aesthetics and Technology of Live Performance of Algorithmic Electronic Dance Music. En: *Proceedings of the International Conference on New Interfaces for Musical Expression*. London, United Kingdom : Goldsmiths, University of London, June 2014, p. 355–358

[24] Collins, Nick ; Mclean, Alex ; Rohrhuber, Julian ; Ward, Adrian: Live coding in laptop performance. En: *Organised Sound* 8, Nr. 3, p. 321–329

[25] Cárdenas, Alexandra: *Street Code - Live Coding in the Public Space*, University of the Arts, Tesis de Grado, 2017

[26] Dannenberg, B: Machine Tongues XIX: Nyquist, a Language for Composition and Sound Synthesis. En: *Computer Music Journal* 21 (1997), Nr. 3, p. 50–60

[27] Dascal, Marcelo. *Colonizing and Decolonizing Minds*

[28] Debray, Regis: *Introducción a la Mediología*. 1st. PaidosIberica, 2001. – ISBN 9788449310263

[29] Feyerabend, Paul: *Tratado contra el método, Esquema de una teoría anarquista del conocimiento*. Tecnos, 2007 ISSN 9788430946082

[30] Flusser, Vilém: *Hacia una filosofía de la imagen*. Trillas : SIGMA, 2010

[31] Gaspar, A ; Langevin, S: Restoring coding with intention in introductory programming courses. En: *SIGITE*, 2007, p. 91–98

[32] Gibb, Alicia M.: *NEW MEDIA ART, DESIGN, AND THE ARDUINO MICROCONTROLLER: A MALLEABLE TOOL*, Pratt Institute, Tesis de Grado, 2010

[33] Herrera Machuca, Mauro ; Lobato Cardoso, Jaime A. ; Torres Cerro, José A. ; Lomelí Bravo, Fernando J.: Live coding for all: three creative approaches to live coding for non-programmers. En: *International Journal of Performance Arts and Digital Media* 12 (2016), Nr. 2, p. 187–194. – ISSN 20400934

[34] Hindle, A: Orchestrating "Your Cloud Orchestra. En: *Proceedings NIME* (2014)

[35] Hoad, F: *The Concise Oxford Dictionary of English Etymology*. Oxford : Oxford University Press, 1996

[36] Kirkbride, R: FoxDot: Live Coding with Python and SuperCollider. En: *Proceedings of the International Conference on Live Interfaces*, 2016

[37] Kostelanetz, R: Live Coding of Consequence. En: *John Cage as a Hörspielmacher* (1990)

[38] Lazzarini, Victor: The Development of Computer Music Programming Systems. En: *Journal of New Music Research* 42 (2013), Nr. 1, p. 97–110. – ISSN 09298215

[39] Magnusson, Thor: *Epistemic Tools: The Phenomenology of Digital Musical Instruments*. UK, University of Sussex, Tesis de Doctorado, 2009

[40] Magnusson, Thor: ixi lang: a Supercollider Parasite for Live Coding. En: *Proceedings of International Computer Music Conference 2011 (ICMC '11)* (2011), p. 503–506

[41] Magnusson, Thor: Herding Cats: Observing Live Coding in the Wild. En: *Computer Music Journal* 38 (2014), Nr. 1, p. 8–16. – ISBN 8187672641

[42] Magnusson, Thor: Interfacing Sound: Visual Representation of Sound in Musical Software Instruments. En: *Musical Instruments in the 21st Century* (2017)

[43] Makela, Mia: The Practice of Live Cinema. En: *MediaSpace* (2008)

[44] Mayo, D.G.: *Error and the Growth of Experimental Knowledge*. University of Chicago Press, 1996 (Science and Its Conceptual Foundations series). – ISBN 9780226511993

[45] Mccartney, James: Continued Evolution of the SuperCollider Real Time Synthesis Environment The Environment Programming. En: *International Computer Music Conference* (1998), p. 133–136

[46] Mclean, Alex: Making Programming Languages to Dance to: Live Coding with Tidal. En: *FARM* (2014)

[47] Mclean, Alex: *Torque #1*. 2014. – 141–144 p.

[48] Mclean, Alex ; Griffiths, Dave ; Vzw, Foam ; Collins, Nick ; Wiggins, Geraint: Visualisation of Live Code. En: *Electronic Visualisation and the Arts* (2010)

[49] McLean, Alex ; Wiggins, G.: Tidal - Pattern Language for the Live Coding of Music. En: *Proceedings of the 7th Sound and Music Computing conference*, 2010, p. 331–333

[50] Ogborn, David: Live Coding Together: Three Potentials of Collective Live Coding. En: *Journal of Music, Technology and Education* 9 (2016), Nr. 1, p. 17–32. – ISSN 17527074

[51] Ogborn, David ; Tsabary, E. ; Jarvis, Ian ; Cárdenas, Alexandra ; Mclean, Alex: Extramuros: Making Music in a Browser-Based, Language-Neutral Collaborative Live Coding Environment. En: *Proceedings of International Conference on Live Coding* (2015)

[52] Ogborn, E. Navarro del Angel Luis Beverly Jamie Betancur E. ; Mclean, Alex: Browser-based Collaborative Projectional Live Coding of Musical Patterns. En: *Proceedings of International Conference on Live Coding* (2017)

[53] Papert, S: *The children's machine, rethinking school in the age of the computer*. New York : Basic Books, Inc, 1993

[54] Pau, Alsina: Introducción al arte digital. (2017), Nov

[55] Polkinghorne, Donald: Practice and the Human Sciences: The Case for a Judgment-Based Practice of Care. En: *SUNY Press* (204), p. 115–125

[56] Popper, Karl R.: *Conjetures and refutations: the growth of scientific knowlege*. New York : NY: Harper Torchbooks, 1965

[57] Puckette, Miller: Machine Tongues XIX: Nyquist, a Language for Composition and Sound Synthesis. En: *Proceedings, Third Intercollege Computer Music Festival* (1997), p. 1–4

[58] Pérez Royo, Victoria. *El giro performativo de la imagen*. 2010

[59] Roads, Curtis: *The Computer Music Tutorial*. Massachussetts : The MIT press, 1996

[60] Roberts, Charles ; Kuchera-Morin, JoAnn: Gibber: Live Coding Audio in the Browser. En: *Proceedings of the International Computer Music Conference* (2012), p. 64–69. ISBN 9780984527410

[61] Roberts, Charles ; Wright, Matthew ; Kuchera-Morin, JoAnn ; Höllerer, Tobias: Gibber: Abstractions for Creative Multimedia Programming. En: *Proceedings of the ACM International Conference on Multimedia - MM '14*, 2014. – ISBN 9781450330633

[62] Rodriguez, Jessica: *Sistemas algorítmicos en las artes y sus procesos compositivos [relación medio-imagen-cuerpo]. Caso de estudio: Altamisa*/*, División de Arquitectura, Arte y Diseño, U. de Guanajuato, Tesis de Grado, 2017

[63] Rohrhuber, Julian ; De Campo, Alberto: Improvising Formalisation — Conversational Programming and Live Coding. En: *New Computational Paradigms for Computer Music* (2009)

[64] Rubin, Marc J.: The effectiveness of live-coding to teach introductory programming. En: *Proceeding of the 44th ACM technical symposium on Computer science education - SIGCSE '13* (2013), p. 651–656. ISBN 9781450318686

[65] Ruthmann, Alex ; Heines, JM ; Greher, GR: Teaching computational thinking through musical live coding in scratch. En: *SIGCSE '10 Proceedings of the 41st ACM technical symposium on Computer science education* (2010), p. 351–355. ISBN 9781605581835

[66] Salomon, Gavriel: *Interaction of Media, Cognition, and Learning*. Routledge, 1994

[67] Sayer, Tim: Cognition and Improvisation: Some Implications for Live Coding. En: *Proceedings of International Conference on Live Coding* (2015)

[68] Schrödinger, Erwin: *What is life?* Cambridge University Press, 1944

[69] Sorensen, Andrew: Impromptu: an interactive programming environment for composition and performance. En: *Proceedings of the Australasian Computer Music Conference*, 2009

[70] Tober, Brad: Creating with Code: Critical Thinking and Digital Foundations. En: *Mid-America College Art Association Conference* (2012)

[71] Trueman, David ; Smalwood, S. ; Wang, Ge: PLOrk: the Princeton Laptop Orchestra, Year 1. En: *Proceedings of the ICMC* (2006), p. 443–450

[72] Wang, Ge: *The ChucK Audio Programming Language; A Strongly-Timed On-The-Fly Environ/Mentaly;*, Princeton University, Tesis de Grado, 2008. – 192 p.

[73] Wang, Ge ; Cook, Perry R.: ChucK: A Concurrent, On-the-fly, Audio Programming Language. En: *Proceedings of International Computer Music Conference* (2003)

[74] Wang, Ge ; Cook, Perry R. ; Salazar, Spencer: ChucK: A Strongly Timed Computer Music Language. En: *Computer Music Journal* 39 (2015), Nr. 4, p. 10–29

[75] Wessel, David ; Wright, Matthew: Problems and Prospects for Intimate Musical Control of Computers. En: *Proceedings NIME* (2001)