DISEÑO DE UN ALGORITMO PARA MEZCLA DE AUDIO TIPO VST BASADO EN DIAGRAMAS DE GIBSON

BELMAN JAHIR RODRIGUEZ NIÑO

UNIVERSIDAD DE SAN BUENAVENTURA BOGOTA

FACULTAD DE INGENIERIA DE SONIDO

BOGOTA D.C.

2011

DISEÑO DE UN ALGORITMO PARA MEZCLA DE AUDIO TIPO VST BASADO EN DIAGRAMAS DE GIBSON

BELMAN JAHIR RODRIGUEZ NIÑO

Proyecto de grado

Asesor

Raúl Rincón

Director de programa Ingeniería de sonido

UNIVERSIDAD DE SAN BUENAVENTURA BOGOTA

FACULTAD DE INGENIERIA DE SONIDO

BOGOTA D.C.

2011

Nota de aceptación:

_____
_____
_____
_____
_____
_____

_____

Firma del presidente del jurado

_____

Firma del jurado

_____

Firma del jurado

Bogotá D.C. 25 de Abril de 2011

Dedicado a Dios y a mis padres, por su apoyo incondicional y constante formación en la perseverancia.

AGRADECIMIENTOS

A Dios por darme la oportunidad de continuar con mi desarrollo profesional, darme la oportunidad de culminar esta carrera con éxito, guiarme en el desarrollo de este proyecto y mantener siempre en mi una visión positiva de paciencia y perseverancia.

A mis padres por su apoyo incondicional en todo lo que he necesitado, su constancia y paciencia en el proceso de mi educación, que me han convertido en una persona con principios tanto éticos como morales y la oportunidad que me dan de compartir este logro con ustedes.

A mi tutor Raúl Rincón, quien ha sido mi guía en el desarrollo de todos mis proyectos en esta carrera, ha mantenido en mi la idea de avance e innovación y por su colaboración en la revisión y redacción del proyecto.

A David Gibson, por compartir toda su experiencia y propuestas en torno a la mezcla de audio, y crear la necesidad de representar útilmente su teoría.

A Ableton Live y Cycling `74 por permitirme trabajar completamente y sin ningún problema el desarrollo de este algoritmo en sus versiones demo.

A todos los usuarios de internet de la página Cycling ´74, por su participación en el foro.

CONTENIDO

pág.

LISTA DE FIGURAS

pág.

INTRODUCCIÓN

La utilización de algoritmos y programas interactivos ha venido desarrollándose en la industria de la producción musical. Día a día se desarrollan un sin número de programas destinados a la mejora, la versatilidad y el desarrollo ingenieril de los usuarios, tanto por empresas dedicadas a la comercialización de software de audio, como a terceras personas que realizan aportes significativos en la creación de software libre. Por tal razón actualmente se cuenta con una elevada calidad en herramientas para el desarrollo de programas de audio interactivos.

La ventaja de manipular el audio de manera digital, permite operar mediante algoritmos características sonoras tales como amplitud, fase y frecuencia, junto con otras opciones, en este caso visuales, con el fin de simular circunstancias físicas del mundo real.

Los actuales software de audio cuentan con extensiones o subprogramas llamadas ¨plug-in¨, desarrolladas para optimizar el trabajo en el momento de realizar una mezcla, estos proporcionan interfaces graficas al usuario que simulan elementos tales como controles y potenciómetros pertenecientes a los hardware de audio reales.

La mezcla es uno de los aspectos fundamentales e igualmente difíciles de realizar en el proceso de producción, debido a que depende de la pericia y experiencia obtenida al realizar dicho proceso. Nosotros nos relacionamos con el sonido de dos maneras: *Sentimos* (y oímos) las ondas sonoras físicas que salen de los altavoces, y *nos imaginamos* la colocación aparente de los sonidos entre los parlantes. El problema es que la mayoría de la gente no diferencia entre las partes individuales que forman una pieza musical grabada. Escuchan un “sonido” general y rara vez separan la mezcla de la música.

Por esta razón aprovechando la facilidad actual en la manipulación del audio en los software de produccion es necesario el desarrollo un algoritmo que permita visualizar por medio de un plug-in el trabajo de mezcla que se esta realizando para alcanzar un mayor nivel desarrollo.

En este proyecto se desarrollará la herramienta que permita la visualización de un mezcla de audio, por medio de la programación de un algoritmo que sea funcional en una estación de trabajo, en este caso, la version 8 de *ableton live*, de manera completa, versatil y con bajo consumo de recursos de la maquina.

El punto de partida en la creación de esta herramienta de ayuda visual, fué tomado de la experiencia de un productor estadounidense de bastante experiencia llamado David Gibson, quien formula una amplia teoria en su libro “The art of mixing” sus propuestas explican el proceso en torno a como realizar una mezcla de audio de manera balanceada en un espacio tridimensional, con la intensión de que los sonidos no se sobrepongan entre sí y se mantengan espaciados adecuadamente en un espacio esterofónico, la propuesta de David Gibson, es la de hacer una idea mental de cómo se distribuyen los sonidos en el espacio, en base a como se perciben auditivamente, expone y explica conceptos desde volumen, como se percibe y se debe mentalizar visualmente, paneo, frecuencia, efectos de tiempo, compresores, y una extensa tematica de todos los aspectos y elementos que influyen en el proceso de mezcla, junto con una amplia de generos musicales que David Gibson a analizado minusiosamente.

El libro de David Gibson “The Art of Mixing” piedra angular que sostiene todo el desarrollo del algoritmo propuesto en este proyecto. Inicialmente en la programación del algoritmo se utilizara el software Max Msp que permite el diseño en el entorno de la estación de trabajo Ableton live, la señal de audio sera tomada de ableton live, para procesarla en Max for Live, se debe programar inicialmente un ecualizador para caracterizar el sonido de cada canal, la salida de la señal de audio del algoritmo ecualizador de entrada continuara a un algoritmo que analice el contenido frecuencial del sonido de cada canal, posteriormente pasara a un algoritmo que modifique el paneo de la señal de audio en el canal seleccionado, luego debe continuar a otro algorimto que permita manipular el volumen en el canal seleccionadao, y finalmente tomar datos numericos de cada uno de los anteriores algoritmos para manipular una ventana donde se visualizara visualmente cada canal en un espacio tridimensional, por medio de una esfera, objeto que sugiere David Gibson para la interpretacion de los sonidos .

1. PLANTEAMIENTO DEL PROBLEMA

1.1 ANTECEDENTES

Cientos de empresas informáticas con diferentes enfoques realizan cada año una gran cantidad de plug-ins destinados al desarrollo del audio, no muchos de estos productos mantienen una visualización de un espacio simulado en los procesos que se están realizando, solo algunos que trabajan con análisis acústico poseen estas características. Como por ejemplo la geometría del recinto en el caso del plug-in AMPHIOTICK ENHACER PR (VST) y RAYSPACE (VST).

Actualmente no se ha desarrollado un plug-in para mezcla que visualice los tracks de audio pertenecientes a una mezcla estereofónica en un espacio tridimensional, la herramienta más viable para el desarrollo de este plugin la proporciona uno de los software de programación de Miller Puckette como MAX/MSP y PURE DATA.

Miller Puckette originalmente escribió Max, en ese entonces nombrado *Patcher*, un editor para el Macintosh en el IRCAM a mediados de los años 80 para que compositores tengan acceso a un sistema de autor de música interactiva hecha con ordenadores. Primero fue usado en una composición de piano y ordenador llamada *Plutón*, sincronizando el ordenador con el piano y controlando un Sogitec 4X, el cual procesaba el audio.

En 1989, IRCAM desarrolló y mantuvo una versión concurrente llamada Max/FTS «Faster Than Sound», y es una analogía a su precursor MSP; este último fue mejorado al implementar el soporte físico DSP al ordenador).

En 1996, Puckette publicó un software libre completamente rediseñado llamado Pd (*Pure Data*), el cual contiene varias diferencias fundamentales con el IRCAM original, sin embargo, sigue siendo un sustituto interesante para aquellos que no quieran gastar cientos de dólares en Max/MSP.

Max tiene numerosas extensiones y aplicaciones; en particular, las extensiones de audio que aparecieron en 1997, trasladadas de Pure Data. Estas fueron llamadas MSP (las iníciales de Miller S. Puckette, autor de Max y Pd). Estas adiciones para Max permitieron que el audio digital sea manipulado en tiempo real, y a la vez, permitía a los usuarios la creación de sus propios sintetizadores y efectos-procesadores. Previamente, Max había sido diseñado con el fin de tener un terreno común con sintetizadores reales, samplers, etcétera, como un «lenguaje de control» usando MIDI o algún otro protocolo de red.

- 1997. The art of mixing. A Visual Guide to Recording, Engineering, and Production. DAVID GIBSON.

En el libro de David Gibson se desarrolla toda la teoría acerca la visualización imaginaria o subjetiva al escuchar y realizar una mezcla, es una guía vital en el desarrollo de un plug-in para interpretación grafica de una mezcla. Mediante la implementación de gráficos en tres dimensiones para identificar a cada elemento que integra la mezcla de una canción, David Gibson, el autor, asigna formas y colores a cada instrumento o sonido, y coloca a dichas formas, de diferentes volúmenes, en alturas distintas dentro de un espacio virtual que representa el campo estéreo entre los parlantes, izquierdo y derecho, de un sistema de audio.

El propósito del libro “The Art of mixing” es que el lector comprenda que cada sonido ocupa un determinado lugar en el espacio virtual de audición y, que al mismo tiempo, ese sitio puede ser compartido por uno o varios sonidos.

En sus más de 300 páginas, se tratan, conceptos sencillos, las herramientas que el técnico o ingeniero de mezcla posee para manipular los elementos que integran la canción. El término manipular es apropiado ya que cada fuente sonora se identifica con una forma específica ubicada dentro del espacio virtual determinado por los parámetros: ancho (panorama); alto (frecuencia); profundidad (volumen).

La idea de manipulación apunta a modificar la forma y ubicación de los sonidos, de la misma manera que un artista lo hace con los elementos que componen su obra, hasta darle la apariencia que desea. Al asimilar los conceptos planteados el lector estará en condiciones de escuchar cualquier canción y distinguir los diferentes componentes y su ubicación, pudiendo luego, aplicar ese conocimiento en su propio trabajo.

El libro contiene numerosas ilustraciones, mostrando gráficos de mezclas de géneros o estilos específicos. Además, el lector encontrará representaciones individuales para cada sonido, como también algunos ejemplos gráficos de mezclas de canciones famosas de Pink Floyd; Peter Gabriel; Norah Jones; Avril Lavigne, entre otros, que seguramente el usuario reconoce inmediatamente.

En la Universidad de San Buenaventura sede Bogotá, se ha trabajado sobre el desarrollo de algoritmos para el procesamiento de audio, como trabajos reconocidos se encuentran: el proyecto de grado del alumno Juan Manuel Medina Sánchez en el año 2006 “Diseño y construcción de sistemas electrónicos virtuales para el procesamiento y la generación de señal musical en tiempo real soportados como plug-ins vst: Vocoder de fase y reverberador digital ”, Proyecto de grado del alumno Joel Leandro Moreno Báez en el año 2006 “Diseño de un dispositivo virtual tipo VST para patrones rítmicos de percusión del Atlántico Colombiano” y finalmente en el año 2009 un grupo de egresados; María Mercedes Ocampo, Jorge Otero y Alexander Campos, realizaron un proyecto de grado llamado “Diseño de un software, mediante un algoritmo matemático, que convolusione una señal de audio, con la respuesta al impulso de recintos cerrados, en la ciudad de Bogotá ”.

1.2 DESCRIPCIÓN Y FORMULACIÓN DEL PROBLEMA

La creación de plug-ins es un gran reto, debido al costo, el aprendizaje de las herramientas y el desarrollo de aplicaciones que intervienen en la adquisición de un software para la creación de extensiones vst, además de esto aun no existe un plug-in que integre efectos mediante una interpretación visual con el fin de tener

más versatilidad y espacialidad en el proceso de mezcla de audio. Por lo tanto la pregunta que representa el reto para el desarrollo de este proyecto es:

¿Cómo diseñar e implementar un algoritmo que permita realizar una mezcla de audio satisfactoria, teniendo una ayuda visual aparte de la sonora, utilizando un entorno tridimensional?

1.3 JUSTIFICACIÓN

Los ingenieros debemos apuntar hacia la realización de proyectos versátiles, completos y útiles para facilitarle trabajo y tiempo a la sociedad, junto con el desarrollo de nuevo software competitivo en el mercado

El diseño de un plug-in tipo VST llevado a cabo como proyecto de grado favorece la interpretación de este tipo de procesos al usuario, ofrece una alternativa opcional, económica y versátil que provea de elementos prácticos para ser empleados en mezcla y masterizacion de audio. Incentivar la mejora del producto y desarrollo audiovisual, además suministra un elemento de investigación para el entorno universitario como ayuda para quienes deseen desarrollar este tipo de aplicaciones.

1.4 OBJETIVOS

1.4.1 Objetivo General

Diseñar y construir un algoritmo que permita visualizar el proceso de mezcla de audio, en un entorno visual tridimensional basado en la teoría de David Gibson.

1.4.2 Objetivos específicos

- Analizar y evaluar las características y propiedades de los efectos o parámetros comúnmente usados en una mezcla estero.

- Estudiar, analizar y concluir los algoritmos que van a ser usados recrear estos parámetros en un software de audio.

- Realizar un algoritmo para la manipulación de 3 parámetros base en la mezcla (ecualización, paneo, volumen)

- Diseñar y construir un algoritmo que permita manipular parámetros básicos en mezcla audio con un objeto “esfera” en un espacio tridimensional independientemente.

1.5 ALCANCES Y LIMITACIONES

1.5.1 Alcances

- Calibración de sistemas de reproducción mediante la comparación espectral de un micrófono de medición con una señal de referencia (ruido blanco)

- Utilizar la convolución de la respuesta al impulso de un recinto deseado, con la señal de audio para visualizar en un espacio tridimensional volumen de la sala.

- Manejo de los parámetros de mezcla, mediante una interacción física por medio de una cámara.

- Generar un aporte significativo en cuanto al desarrollo de software de producción de audio interactivo.

1.5.2 Limitaciones

- Desarrollo de plugins solamente para Ableton Live 8 y posteriores.

2. MARCO DE REFERENCIA

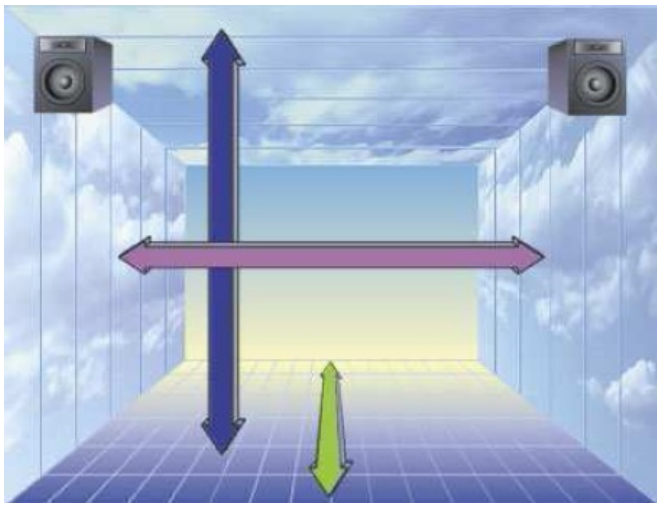
2.1 MARCO TEÓRICO – CONCEPTUAL

Estructurado de manera consecuente acerca de el proceso de realización del software o plugin, aclarando primeramente como seria la posible representación visual de la mezcla, que algoritmos manejar para esta representación y como diseñarlos.

2.1.1 Representacion Visual Del Espacio Sonoro.

Primero se definirá el espacio tridimensional, las características de los ejes de simetría con relación al sonido. Hay 3 parámetros básicos del sonido que corresponden a los ejes visuales X, Y, y Z.

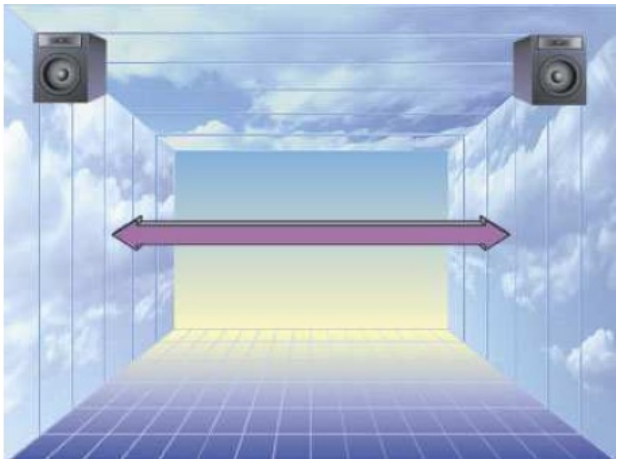
Fig. 1. Ejes de simetría del espacio sonoro.



2.1.1.1 Paneo (eje X).

El paneo, la colocación izquierda/derecha de sonidos en el eje horizontal X, es naturalmente mostrada como la colocación mas a la izquierda o a la derecha de los elementos[1].

Fig. 2. Eje x, Representación de Paneo

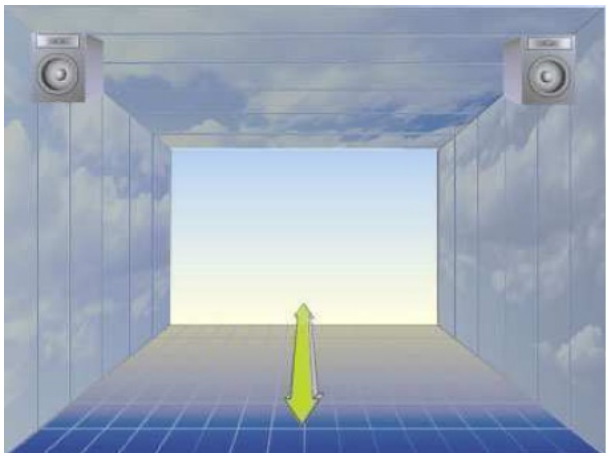


2.1.1.2 Volumen (eje Y).

Los sonidos que están más cerca nuestro son más fuertes y los sonidos distantes son más suaves, por lo tanto, el volumen de un sonido en la mezcla puede ser asignado más adelante o más atrás sobre el eje de simetría Y. Creando así el eje X y Y el semiplano paneo volumen.

[1] GIBSON, David; The Art of Mixing, 1997 pág. 44.

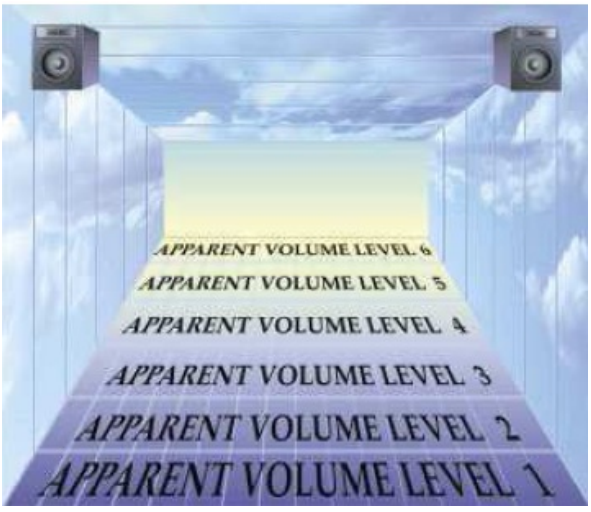
Fig. 3. Eje Y, representación de volumen



Normalmente voces e instrumentos principales se encuentran adelante mientras que otros instrumentos, como cuerdas y voces de fondo, están a menudo en el fondo por eso el termino “voces de fondo.”

El volumen aparente de un sonido también depende de la forma de onda del sonido. Por ejemplo, una sierra eléctrica sonara más fuerte que una flauta, incluso si ambas están exactamente al mismo volumen. El volumen aparente es el nivel que los sonidos parecen estar para nuestros oídos.

Fig. 4. Niveles aparentes de volumen



Si pensamos el volumen en decibeles, basados en el nivel de presión del sonido, entonces un sonido podría estar en 140 niveles diferentes de volumen en una mezcla. Pero para hacer este amplio rango de niveles mas manejable, se dividiran en 6 niveles diferentes, o capas, de adelante hacia atrás, uno es el mas fuerte y seis el mas suave.

2.1.1.2.1 Aparente NIVEL 1 de Volumen

Los sonidos en este nivel son escandalosamente fuertes. A decir verdad, es muy raro e inusual, que ubicar sonidos en este nivel. Comúnmente, solo sonidos que duran muy corto tiempo a este nivel. En este nivel se suelen colocar sonidos con gran contenido energético en corto tiempo, “impulsivos” explosiones, gritos, y otros efectos especiales también pueden estar tan fuertes.

2.1.1.2.2 Aparente NIVEL 2 de Volumen

Los principales sonidos a este volumen son las voces principales y los instrumentos principales. Rap, hip-hop, y música dance a veces tienen el bombo en este nivel. Este nivel realmente parece muy fuerte en la mezcla, y se usa en la música donde las voces o la letra son el foco principal de atención.

2.1.1.2.3 Aparente NIVEL 3 de Volumen

Los sonidos de este nivel son las partes principales del ritmo, como batería, bajo, guitarra, y teclados. Las voces principales de algunos rock-n-rolles cuando se ajusta atrás en la música.

2.1.1.2.4 Aparente NIVEL 4 de Volumen

Los sonidos a este volumen incluyen colchones de ritmo, y pads de acordes, como pianos de fondo, teclas, o guitarras. La batería de jazz, y rock fácil a menudo esta puesta a este nivel. Voces de fondo, cuerdas, y reverb también son puestas a menudo aquí.

2.1.1.2.5 Aparente NIVEL 5 de Volumen

Los sonidos en este bajo nivel no son muy claros y distinguibles. Ellos incluyen al sonido del bombo de batería en el jazz y en la música de big band. Muchos efectos y la reverb son a menudo colocados aquí para que se escuchen solo cuando oyes atentamente. Las voces de fondo a veces están relegadas a este nivel. Otros instrumentos son a veces puestos a este nivel solo para rellenar la mezcla.

2.1.1.2.6 Aparente NIVEL 6 de Volumen

Los sonidos ubicados tan atrás en la mezcla son tan suaves que son difíciles de detectar. Esto también incluye los mensajes subliminales y el enmascaramiento de fondo (sonidos que suenan muy atrás). Si estos sonidos no son ajustados de la forma correcta, podrían ser percibidos como ruido[2].

Tabla. 1. Niveles aparentes de volumen.

NIVEL 1	NIVEL 2	NIVEL 3	NIVEL 4	NIVEL 5	NIVEL 6
Despertador	Voces principales	Ritmo Principal	Colchón de Ritmo	Efectos	Susurros
Explosiones	Instr. Principales	Voz Principal	Pads de Acordes	Bombo (Jazz)	Conversaciones
Gritos	Hits de Horns	Toms	Batería (Jazz)	Murmullos	Ruidos
	Hits Sinfónicos	Snare (Dance)	Voces de Fondo	Voces de Fondo	Doubling (Doblar)
		Bombo (Metal)	Cuerdas		
		Hi-Hat (Jazz)	Reverb		
		Efectos Fuertes			

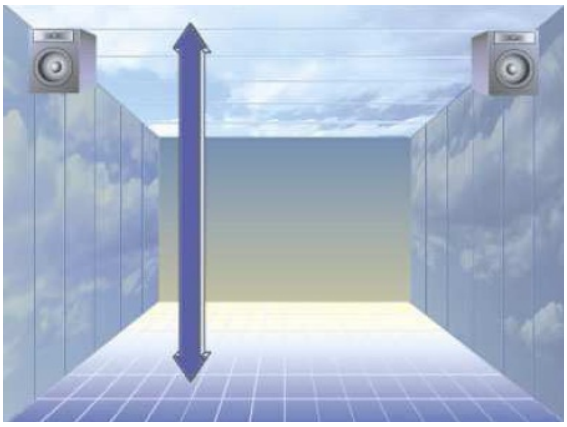
[2] Gibson. David, “The art of mixing”, 1997. pág 197.

Los niveles de volumen aparte de caracterizar el eje Y del espacio tridimensional, también afectara el radio de los elementos, mas adelante se explicaran estas variantes.

2.1.1.3 Frecuencia (eje Z).

Hay una ilusión interesante que ocurre con las frecuencias altas y bajas la representación del espacio sonoro. Las frecuencias serán la característica que variara la ubicación de los elementos en el eje Z, ubicando así frecuencias altas en lugares altos y bajas en lugares bajos del espacio. Instrumentos como campanas, platillos o cuerdas altas siempre parecen ser mucho más altas en el espacio que otros instrumentos, como bajos, bombos de batería, y bombos de rap[3].

Fig. 5. Eje Z, representación de frecuencia.



Hay varias razones para que exista esta ilusión. Primero que nada, las frecuencias bajas llegan a trabes del piso a tus pies; las frecuencias altas no. A decir verdad, los estudios profesionales están calibrados para saber exactamente cuantas frecuencias bajas viajan a través del piso hasta tus pies.

[3] GIBSON, David; The Art of Mixing, 1997 pág. 47.

Un sonido puede ser llevado por todas partes en el espacio cambiando el volumen, el paneo, y el tono. Si el enmascaramiento es un asunto grande, entonces es importante saber cuanto espacio un sonido ocupa en el limitado mundo de la representación del espacio sonoro. A decir verdad, no solo diferentes sonidos ocupan más o menos espacio, la ecualización, y los efectos pueden hacer una diferencia enorme respecto a cuanto espacio un sonido usa.

2.1.2 Representacion Visual De Los Sonidos.

¿Que tamaño ocupa el sonido en el entorno 3D?

Con el espacio limitado, se necesita saber el tamaño de cada miembro de la multitud para contribuir al gran problema del enmascaramiento. Cuanto mas espacio ocupe el sonido, mas enmascarara a otros sonidos en la mezcla[4].

2.1.2.1 Tamaño en función de la frecuencia.

Sonidos graves, siendo mas grandes, enmascaran mas a otros sonidos. Las altas frecuencias ocupan menos espacio en el entorno de imágenes. Por lo tanto, las representaciones visuales de los sonidos de alta frecuencia, son más pequeñas y puestas más altos que los instrumentos de baja frecuencia, que son representados por grandes imágenes ubicadas abajo entre los parlantes.

2.1.2.2 Tamaño en función del volumen

Cuanto mas fuerte esta un sonido en la mezcla, mas enmascara a otro sonido. Por lo tanto, sonidos más fuertes son más grandes. Cuanto mayor sea el volumen el radio de los elementos también será mayor por esta razón el volumen afectara tanto la profundidad del espacio como el tamaño de los elementos.

[4] GIBSON, David; The Art of Mixing, 1997 pág. 56.

2.1.2.3 Forma

Se puede pensar al principio, que un “punto” entre el entorno visual puede parecer apropiado. Cuando un sonido, como una voz, es paneada a la izquierda, el punto se puede mover para lado izquierdo. Esta es una común representación usada por mucha gente cuando discuten la ubicación derecha/izquierda de los sonidos en el campo estéreo.

Una imagen redonda es más apropiada, especialmente cuando se considera la forma en que dos sonidos parecen coincidir cuando son paneados para la derecha o izquierda desde el centro. Cuando son agregados juntos y empieza la superposición en el medio, la imagen sugiere que los sonidos deben ser redondos y simétricos.

Un punto sólido tendría sus defectos. Los sonidos no son objetos sólidos. Dos sonidos pueden estar en el mismo lugar en la mezcla y aun ser escuchados claramente por separado. Por lo tanto, tiene sentido hacer los sonidos transparentes. Si utilizas esferas transparentes para representar el campo del sonido en la imagen que creamos entre los parlantes, entonces dos sonidos pueden ser vistos y escuchados en el mismo lugar.

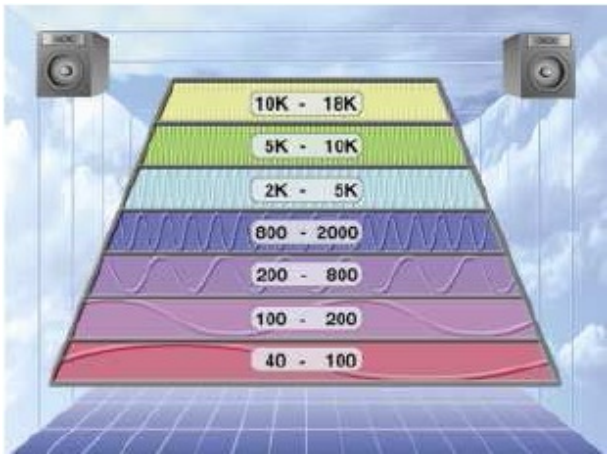
Enmascaramiento, cuando un sonido tapa u oscurece a otro sonido, es el mayor problema en la mezcla. Si tienes dos sonidos en el mismo lugar entre los parlantes, uno de los sonidos a menudo será escondido por el otro. Por que el espacio entre los parlantes es limitado y el enmascaramiento es el mayor problema en la mezcla, el asunto general de la mezcla, se vuelve un…control de multitudes.

2.1.2.4 Color

La función principal del color es para diferenciar diferentes tipos de sonidos. Diferentes colores podrían corresponder a diferentes colores del sonido, tipos de

formas de onda, o rango de frecuencia[5]. En este caso se usara para diferenciar los rangos de frecuencia en los que el sonido se encuentra.

Fig. 6. Relación Color - Frecuencia



En conclusión la variación de la ecualización provocara, cambios del tamaño de los elementos, cambios en la posición vertical del espacio y cambios en el color de los objetos.

2.1.3 Programador Para El Diseño Del Algoritmo.

La herramienta mas viable para la realización del algoritmo es el programa Max de la empresa Cycling '74, junto con su versión libre Max for Live el cual permite manejar un lenguaje de programación orientada a objetos en tiempo real.

2.1.3.1 Max/Msp

Es un entorno de desarrollo gráfico para música y multimedia desarrollado y mantenido por Cycling '74, una empresa de programas situada en San Francisco[6].

[5] GIBSON, David; The Art of Mixing, 1997 pág. 72.
[6] http://es.wikipedia.org/wiki/Max_(programa).

El programa ha sido usado durante más de quince años por compositores, artistas y diseñadores de programas interesados en la creación de programas interactivos.

Max es bastante modular, y la mayoría de las rutinas forman parte de una biblioteca compartida. La IPA (Interfaz de Programación de Aplicaciones) permite el desarrollo de nuevas rutinas (llamadas «objetos externos») por terceras personas. Por consecuencia, muchos de los usuarios de Max son programadores no afiliados a Cycling '74 que mejoran el programa, creándole extensiones comerciales y no comerciales. Debido a su diseño extensible e interfaz gráfica considerado por muchos como la lengua franca para el desarrollo de programas de música interactiva.

En el 2003, la segunda adición para el Max/MSP, llamada *Jitter*, fue publicada, agregándole la posibilidad de procesar vídeo, 3D, y matrices en el programa. 2 años más tarde se incorporan al software los objetos *mxj* y *js*. Estos objetos son "externals" que ejecutan código Java y javascript en Max.

Ableton y Cycling '74 han trabajado en conjunto para desarrollar una versión de Max creada para extender las posibilidades de Ableton Live agregándole nuevos elementos. Max for Live estará capacitado para crear todo tipo de dispositivos como instrumentos, efectos, secuenciadores por pasos y herramientas de todo tipo.

Características principales:

- Controles nativos de Live para la creación de todo tipo de herramientas.
- Posibilidad de diseñar y crear nuevos agregados al estilo Live, manejando sus mismos objetos lo que indica que cuando el usuario diseñe algún dispositivo en Max, en Live se mostrará con el mismo diseño y la interfaz característica de Ableton.
- Integración con varias características de Live como la función deshacer, el sistema de automatización MIDI, el manejo de colores o las descripciones para los puntos de información.
- Gran cantidad de objetos varias funciones y controles

- Contiene algunos ejemplos como un secuenciador por pasos, una herramienta para el tratamiento del buffer y un dispositivo de manipulación de loops[7]

Muchos usan Max, aunque no se den cuenta de ello. Los documentos de Max (llamados «patchers») pueden juntarse para formar aplicaciones standalone y pueden distribuirse gratuitamente o ser vendidas.

Con la versión 8 de Ableton Live, ha llegado una oportunidad que parece crear un puente entre este mundo de profesionales de Max/MSP y los del dj-loop aficionados, ganando por ambos lados; Ableton Live 8 constituye un sistema muy sencillo que puede ser utilizado como un reproductor, un instrumento virtual y/o un procesador de señal para gestionar unas cuantas cosas relativamente sencillas en vivo. Además tiene una cosa que en Max/MSP resulta desafortunadamente muy mal implementada: una línea de tiempo. Por otro lado Max/MSP presenta todas las posibilidades de un sistema abierto; la unión de estos dos sistemas, con la posibilidad de usar Ableton Live como plataforma dentro la cual utilizar Max/MSP además de los DSP, sampler y sintetizadores internos del programa, nos permite realmente disfrutar de las cualidades de ambos. Es un discurso de pura economía y esfuerzo: ¿por qué perder tiempo en crear una reverb si la tengo ya hecha y funcionando, y puedo también controlarla con Max/MSP quitando las restricciones que los plug-ins predeterminados nos imponen? Y por qué no disfrutar de la libertad que Max/MSP nos oferece, usando una cómoda y bien hecha *timeline* de audio y video presente en Live. Es un motivo más para acercarse a este programa[8].

Jitter es una colección de objetos de max para la reproducción y manipulación de vídeo, operaciones matriciales y open gl. Por otro lado, jitter nos permite aprovechar al maximo las libererias Open Gl. Open Graphic Library es una libreria escrita por Silicon Graphics pensada en un principio para uso exclusivo con sus maquinas, pero que hoy en dia se ha convertido en una de las mas importantes por su caracter multiplataforma. Esta libreria contiene una serie de comandos para comunicarse directamentre con el chip de nuestra tarjeta grafica y explotar sus posibilidades en el trabajo de renderizado a tiempo real con graficos 3d.

[7] http://www.hispasonic.com/noticias/cycling-74-ableton-presentan-max-for-live/3554
[8] Colasanto. Francisco, “Max/Msp: guía de programación para artistas”, 2009. Pág 25.

2.2 MARCO LEGAL O NORMATIVO

Licencias de software. Las licencias son contratos entre el dueño del programa y la persona o empresa que lo quiere adquirir (usuario). Generalmente, en el contrato se estipulan las cláusulas, términos, posibilidades y restricciones que la empresa dueña del producto la utilizan para con el comprador. Este contrato se hace con respecto a las leyes de propiedad intelectual y derechos de autor. Los tipos de licencia se encuentran a continuación:

Software libre. Es aquel en el cual el usuario tiene libertad de utilizar el programa para cualquier objetivo o propósito. Además tiene acceso al algoritmo o fuente para alterarlo según sus necesidades. Puede distribuir copias gratuitas a cualquier usuario que quiera obtener el software[9].

Software de fuente abierta. Sus términos de distribución cumplen los criterios de distribución libre, inclusión del código fuente, permitir modificaciones y trabajos derivados en las mismas condiciones que el software original, integridad del código fuente del autor, pudiendo requerir que los trabajos derivados tengan distinto nombre o versión, no discriminación a personas o grupos, sin uso restringido a campo de actividad, los derechos otorgados a un programa serán válidos para todo el software, redistribuido sin imponer condiciones complementarias, la licencia no debe ser específica para un producto determinado y no debe poner restricciones a otro producto que se distribuya junto con el software licenciado, la licencia debe ser tecnológicamente neutral.

Estándar abierto. Según Bruce Perens, es basado en los principios de disponibilidad, maximizar las opciones del usuario final, sin tasas sobre la implementación, sin discriminación de implementador, permiso de extensión o restricción, evitar prácticas predatorias por fabricantes dominantes.

[9] http://www.acis.org.co/index.php?id=319

Copyright. Forma de protección proporcionada por las leyes vigentes en la mayoría de los países para los autores de obras originales incluyendo obras literarias, dramáticas, musicales, artísticas e intelectuales, tanto publicadas como pendientes de publicar.

Software de dominio público. Aquel que no está protegido con copyright.

Software con copyleft. Software libre cuyos términos de distribución no permiten a los redistribuidores agregar ninguna restricción adicional cuando lo redistribuyen o modifican. La versión modificada debe ser también libre.

Software semi libre. Aquel que no es libre, pero viene con autorización de usar, copiar, distribuir y modificar para particulares sin fines de lucro.

Freeware. Se usa comúnmente para programas que permiten la redistribución pero no la modificación (y su código fuente no está disponible).

Shareware. Software con autorización de redistribuir copias, pero se debe pagar cargo por licencia de uso continuado.

Software privativo. Aquél cuyo uso, redistribución o modificación están prohibidos o necesitan una autorización.

Software comercial. El desarrollado por una empresa que pretende ganar dinero por su uso.

Dado que este proyecto fue desarrollado bajo un entorno de desarrollo de licencias de software semi libre “demo” y freeware, su clasificación cabe en este tipo de licencia.

En este proyecto no se implementó un marco legal o normativo basado en legislación colombiana de software, ya que no hay una normativa clara con respecto a la legalización de software.

3. METODOLOGÍA

3.1 ENFOQUE EMPÍRICO ANALÍTICO

El enfoque de la investigación es empírico analítico ya que se toma la señal de audio proveniente de un track, y luego por medio de un algoritmo se analiza la amplitud, la frecuencia y la fase utilizando la transformada rápida de Fourier, colocando a disposición del usuario algunos controles para el posicionamiento de las representaciones espaciales de cada canal.

3.2 LINEA DE INVESTIGACIÓN DE USB / SUB-LINEA DE FACULTAD / CAMPO TEMATICO DEL PROGRAMA.

Este proyecto pertenece al campo de investigación de Diseño de Sistemas de Sonido, debido al desarrollo de un programa tipo plug-in el cual se podrá emplear en una plataforma para audio comercial.

La sub-linea de investigación de la facultad es la de Procesamiento de Señales Digitales y/o analógicas, ya que se emplearan señales de audio para ser manipuladas y analizadas, incluyendo funciones digitales para el desarrollo del trabajo propuesto.

La línea de investigación pertenece a Tecnologías actuales y sociedad, puesto que lograra hacer un aporte en ingeniería de sonido, producción musical, programadores de software de audio y todo aquel interesado en el campo del audio digital.

3.3 TÉCNICAS DE RECOLECCION EMPLEADAS

La recolección de información, se realiza básicamente con la búsqueda encontrada en bibliotecas, Internet y cualquier recurso de aplicación de software (manual). Los elementos a utilizar, son los software MAX/MSP (versión demo), Live ableton suite 8 (versión demo).

3.4 HIPÓTESIS

En la investigación se estudiará y comprobará la posibilidad de la creación de un plugin funcional en live 8 que sea capaz de visualizar tridimensionalmente la posición y características de las pistas de audio en una mezcla estéreo, aplicando conocimientos ingenieriles en la programación y diseño del mismo.

3.5 VARIABLES

3.5.1 Variables Independientes

- Funciones u objetos empleados en la programación.
- Propiedades de la ventana de visualización grafica.
- Selección del programa para el desarrollo del plug-in (pure data o max/msp).

3.5.2 Variables Dependientes

- Funcionalidad del producto
- Recarga del hardware por uso de alto contenido grafico.
- Estabilidad de la plataforma
- Necesidad de software live 8.

4. DESARROLLO INGENIERIL

A continuación se explicará detalladamente el proceso para la realización del proyecto en mención teniendo en cuenta los objetivos y los conceptos nombrados al canal deseado, automáticamente, posteriormente se abrirá otra ventana, en la cual se visualizara un objeto “esfera”, el cual representará el sonido característico previamente, se expondrán los algoritmos pertenecientes a cada uno de los parámetros expuestos en el Marco Teórico, y su programación en *Max Msp* para *Live* (Versión Demo).

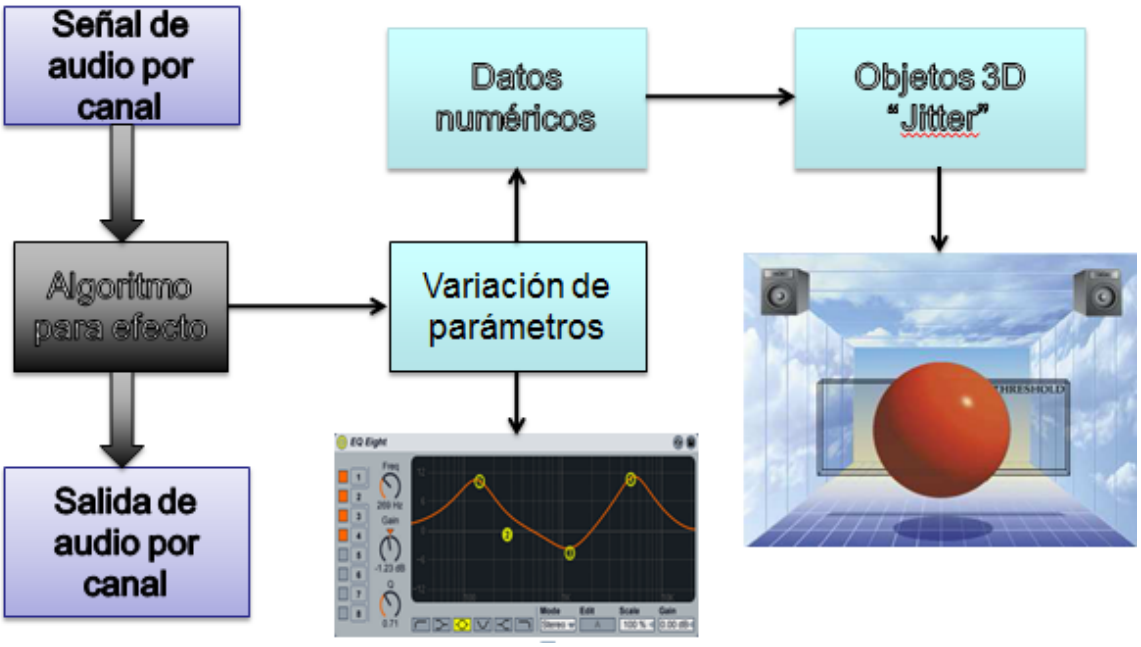
El funcionamiento del proyecto está adecuado para que el usuario asigne el plug-in del canal, por tal razón, el usuario debe asignar un número de *Plug-Ins* a un número de canales respectivamente, visualizándose todos en una sola ventana.

4.1 DESCRIPCIÓN MÉTODO DE REALIZACION DEL PROYECTO

Los algoritmos realizados fueron desarrollados en *Max Msp* para *Live* (Versión Demo), esta es una unión entre los software de producción de audio *Live 8* de Ableton y el entorno de programación *Max Msp 5.1*, cabe anotar que ambos se encuentran en un periodo de prueba, válido por 30 días.

El orden lógico de programación para el proyecto, se observa en el siguiente diagrama de bloques. Este algoritmo será aplicado por canal independientemente ya que cada canal representara un sonido diferente con su respectiva ubicación espacial.

Fig. 7 Diagrama de bloques del algoritmo general.



En el diagrama anterior, la señal de audio que se tiene en el entorno de *Live 8*, entra por medio de un patch a la plataforma de programación de *Max Msp* para *Live*, en esta parte, se va a procesar la señal de audio para adecuarla a la visualización que David Gibson recomienda en su libro. La señal de audio, en el entorno de *Max Msp* para *Live*, permite transformarla a datos numéricos, para ser manejables según las necesidades del usuario, adicionalmente *Max Msp* para *Live*, cuenta con *Jitter* incluido, el cual es el subprograma que permite el manejo de todo el entorno visual.

Finalmente, al tener todo el algoritmo realizado, la señal de audio vuelve al entorno de *Live* para ser manipulada desde allí, sin necesidad de entrar cada vez al entorno de *Max Msp.*

En la realización inicial para enlazar *Max Msp* con *Live*, únicamente se deben tener las últimas versiones desde *Live 8* en adelante, y *Max Msp* 5.1, al tener instalados estos programas, automáticamente al abrir *Live* reconocerá los *Plug-Ins* realizados en *Max Msp* y renombrándolos con una nueva extensión ".amxd" la cual solo funcionará dentro de la plataforma de *Ableton Live 8* en adelante.

El programa inicial para empezar a trabajar en conjunto con *Max Msp*, se abre mediante una herramienta que posee *Live 8* al tener instalado que habilita una

serie de *Plug-Ins* funcionales, el primero de ellos, utilizado para recibir el audio y empezar el desarrollo del algoritmo, se realiza por medio del programa *"Max Audio Effect"* ubicado en la carpeta *"audio effect".*

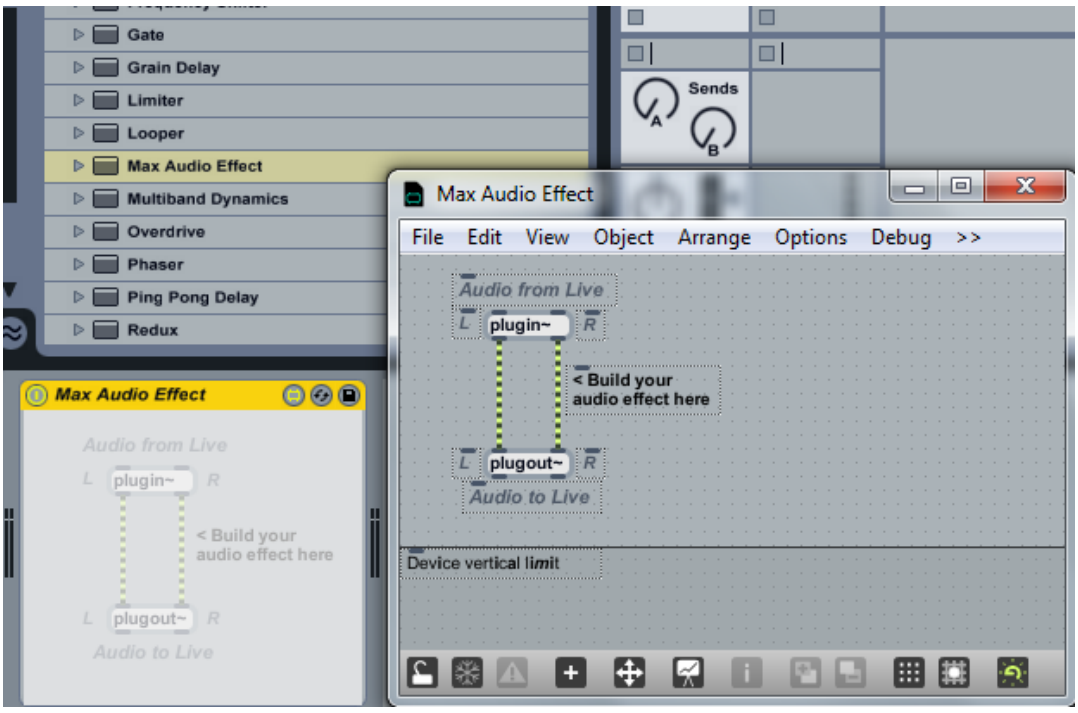
Fig. 8. Ubicación del patch inicial en Live.

Al abrir esta ventana, en la parte inferior de *Live 8*. Aparecerá la configuración interna del Plug-In ó la presentación de esta configuración dentro de *Live*, la función es comunicar el entorno de *Live* con el de *Max Msp*, para iniciar la programación.

Fig. 9. Patch inicial de Max Msp visto en Live.

En Fig.9 sobre la parte superior derecha se observa un ícono el cual abre instantáneamente el entorno de programación de *Max Msp* para *Live*.

Fig. 10. Patch inicial en el entorno de programación Max Msp.

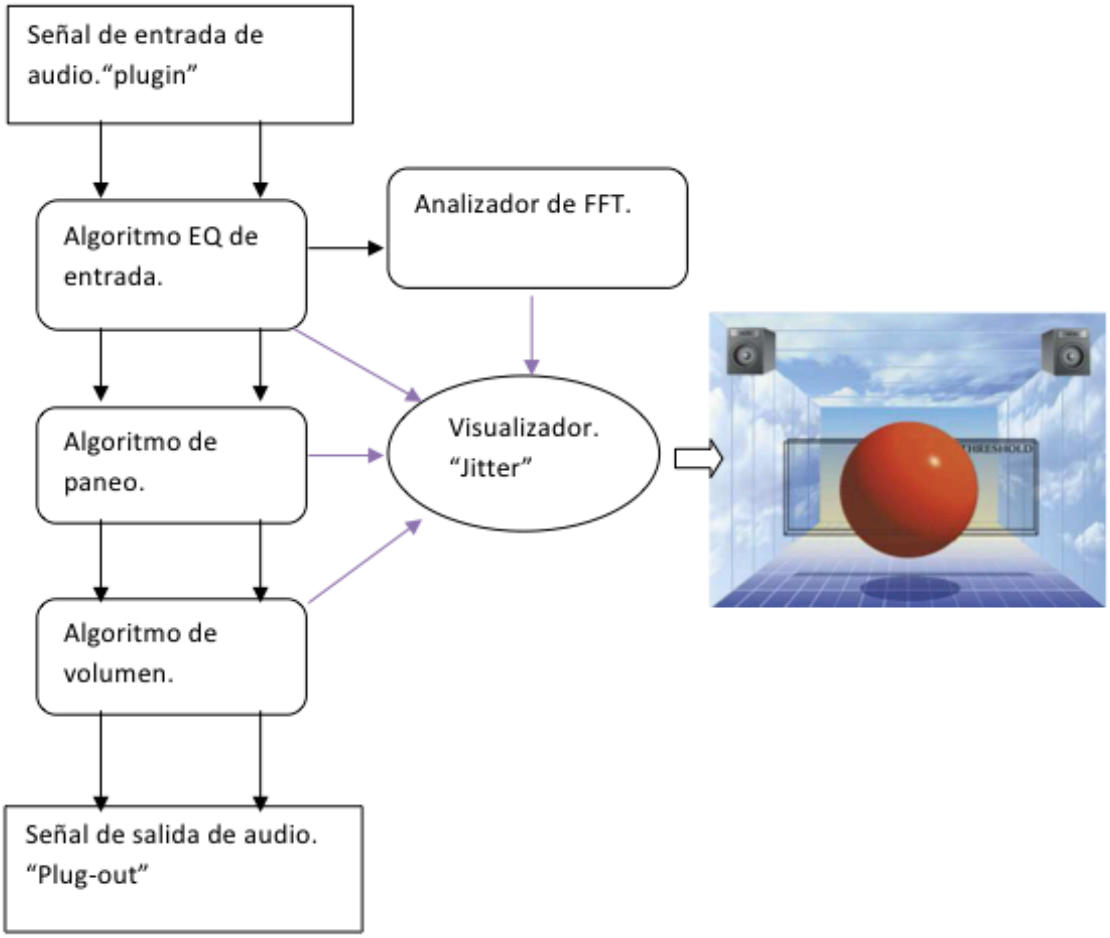


Después de abrir la ventana para el inicio de programación de los algoritmos en Max for live, como se observa en la figura 10. Se procede a diseñar un esquema en el que van a ir los bloques que integran el algoritmo general, estos bloques o

partes del algoritmo general son: Señal de entrada de audio, Algoritmo EQ de entrada, analizador de FFT, Algoritmo de paneo, Algoritmo de volumen, visualizador Jitter y salida de señal de audio

La lógica para el diseño interno del algoritmo dentro de *Max Msp*, se explica a continuación, detallando el siguiente diagrama de flujo de señal de audio, para su posterior procesamiento.

Fig. 11. Diagrama flujo de señal del algoritmo general



A continuación se explicará el proceso de realización de los algoritmos de proceso de señal de audio, los cuales son: Algoritmo Ecualizador de entrada, Algoritmo de FFT, Algoritmo de paneo, y algoritmo de volumen.

Finalmente se explicará el proceso de creación del Visualizador *Jitter*, el cual contiene secciones de cada uno de los algoritmos anteriores.

La señal estéreo que viene por canal ingresa al entorno de programación de *Max Msp* por medio del objeto *Plug-In*. Esta señal ingresa a un algoritmo de ecualización de entrada, este, tiene la función de permitir al usuario caracterizar el sonido por medio de un ecualizador de una banda, un ejemplo claro sería el de un Bajo, este debe procesarse acuerdo a la naturaleza sonora del instrumento.

Como primer algoritmo se encuentra la ecualización, esta va a cambiar el sonido y por lo tanto el tipo de señal para los otros procesos, de esta parte, igualmente sale una señal monofónica hacia un analizador de frecuencia, este tiene la función de recibir la señal de audio ya ecualizada para asignarle un color definido dependiendo también de la naturaleza sonora del canal.

Posteriormente la señal estéreo pasa al algoritmo de paneo, este permite recibir la señal para ser modificada por el usuario en el espacio estereofónico. La señal de salida de este algoritmo ya vendrá con proceso de ecualización y paneo.

El algoritmo final va a variar el nivel ó volumen del canal, de manera estereofónica, para finalmente devolver la señal de audio ya procesada al entorno de *Live.*

Todos estos algoritmos implican los cambios a algunos objetos de Jitter, los cuales son los encargados de convertir la información de audio en información de video.El Visualizador Jitter, es un algoritmo que toma datos numéricos de cada uno de los anteriores algoritmos y los interpreta visualmente en una ventana tridimensional con objetos “esferas”, del algoritmo ecualizador de entrada toma datos que modifican la posición vertical, del algoritmo analizador de fft, obtiene datos que modifican el color de las esferas, del algoritmo de paneo obtiene datos que modificaran la posición horizontal de las esferas y del algoritmo volumen recibe datos numericos que varian la posición de profundidad en el espacio tridimensional

4.2.1. Diseño Del Algoritmo Ecualizador De Entrada.

La función de este bloque, es la de caracterizar el sonido proveniente del canal, el usuario dispone de un ecualizador paramétrico de una sola banda, seleccionable entre: *lowpass, highpass, bandpass, bandstop, peaknotch, lowshelf, highshelf, resonant y allpass*. Por medio de este ecualizador se empezará a dar una ubicación espacial al canal, ya que esta característica frecuencial varía la posición vertical del canal en la ventana de visualización.

Los controles de interfaz para usuario del ecualizador, además de permitir seleccionar el tipo de filtro, incluye un botón de activo e inactivo, permite seleccionar la frecuencia afectada desde 50 Hz, hasta 18 KHz, un selector de ganancia para la frecuencia deseada en rango desde 0dB hasta 30 dB, un factor Q desde su valor mínimo 0.10 hasta 10, un botón de *Reinicio (Reset)*, el cual retorna todos los valores por defecto, y finalmente un *Nivel Maestro (Master)* para asignar una ganancia a la señal total de salida con un deslizador.

Fig. 12. Diseño del ecualizador de entrada en Live.

El plug-in de ecualizador de entrada debe aparecer en el entorno de live como se ve en la Fig. 12 conteniendo las siguientes características:

1. Frecuencia, rango (50 Hz - 18 kHz)

2. Botón activación, (amarillo “activo”).

3. Gain, rango (0dB – 30dB).

4. Q, rango (0.1 - 10).

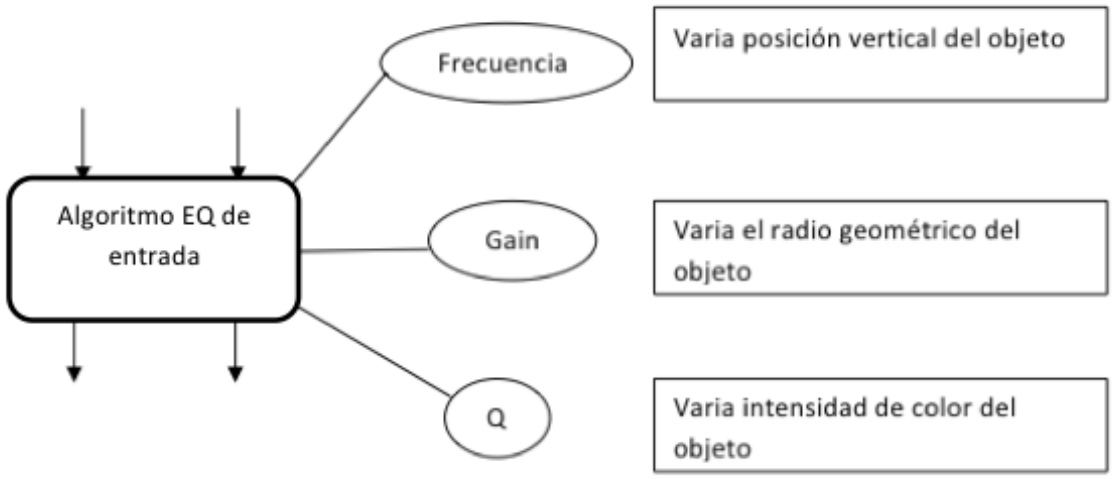
5. Menú desplegable para tipo de filtro (*lowpass, highpass, bandpass, bandstop, peaknotch, lowshelf, highshelf, resonant, allpass*).

6. Reset.

7. Master, Nivel de salida.

El funcionamiento del ecualizador de entrada se explica a continuación junto con su relación con la visualización tridimensional.

Fig. 13. Diagrama flujo de señal ecualizador de entrada.



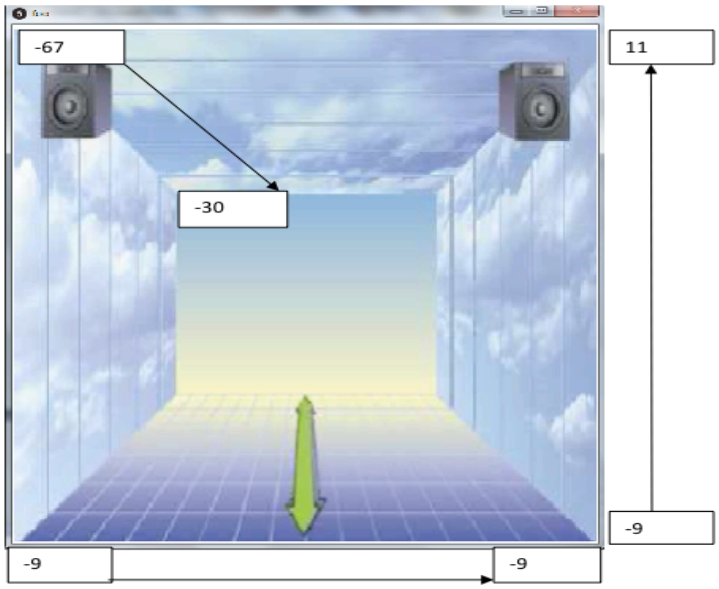
Los valores que el usuario manipule en la interfaz del *Plug-In* en *Live*, varían los parámetros internos del programa en *Max Msp* para *Live*, es decir, si el usuario modifica la frecuencia en la interfaz del ecualizador, este parámetro genera una variación en la posición vertical del objeto “esfera” en la ventana de visualización, como dice la teoría de David Gibson, cuan mayor sea la frecuencia se tendrá una percepción de mayor altura. Por otra parte, el parámetro *Gain*, genera un cambio en el radio del objeto “esfera”, debido a que la ganancia aumenta el nivel de esta ecualización, la cual ocupa mayor lugar espacialmente, y finalmente el factor Q,

determina el ancho de banda ó intensidad en la frecuencia deseada, por esta razón este parámetro se asocia a la intensidad del color en los objetos “esfera”.

Se debe tener en cuenta, que los objetos a visualizar se encuentran en una ventana tridimensional de escalas lineales en los tres ejes de simetría, por esta razón, la frecuencia debe pasar primero por un proceso de linealización, ya que viene en escala logarítmica.

Antes de pasar los valores frecuenciales a los objetos de visualización, se debe aplicar una función logarítmica por medio del objeto (expr log ($f1)) que se puede observar en la figura 35. Este objeto opera el logaritmo en base diez, y su resultado concluirá en dato flotante, a los valores frecuenciales, para ser manipulables en escalas lineales, por esta razón los valores de frecuencia corresponden de la siguiente manera: Para 50 Hz – 3.9129 y para 18 Khz – 9.7981, teniendo estos valores ya en escala lineal, se debe tener en cuenta que la ventana de visualización , tiene unas dimensiones definidas, por esta razón se deben acomodar estos valores lineales de la frecuencia a las dimensiones verticales de la ventana de visualización, a continuación se mostrarán las dimensiones de esta ventana.

Fig. 14. Valores limite de las dimensiones tridimensionales en la ventana de visualización.



Por esta razón para una frecuencia de 50 Hz, corresponderá una asignación de -9 y para una frecuencia de 18 kHz una asignación de 11 en la ventana de visualización.

4.2.2. Construccion Del Algoritmo Ecualizador De Entrada.

Para construir este algoritmo se debe partir de la configuración inicial, el cual recibe la señal de audio de *Live*, para ser procesada en *Max Msp* y posteriormente devuelta al programa que hospeda la aplicación para su edición, esta configuración inicial contiene únicamente dos objetos: *Plug-In* y *Plug-Out*, en la parte media entre estos objetos se construye el algoritmo ecualizador de entrada.

A continuación se muestra el desarrollo del ecualizador de entrada de una banda, en el cual se explica, el funcionamiento de cada uno de los objetos necesarios para su construcción.

4.2.3 Objetos Utilizados En El Algoritmo Ecualizador De Entrada.

A continuación se muestran los objetos utilizados en el algoritmo ecualizador de entrada, se explica su funcionamiento y rol dentro del programa. Estos objetos se encuentran en la ventana de edición de *Max Msp* para *Live*

Para ubicar un objeto, únicamente se hace doble clic sobre la ventana de *Max Msp* para *Live* y se despliega el siguiente menú.

Se selecciona el objeto deseado y si no se conoce, se selecciona el primero, este se encuentra en la parte superior izquierda, y posteriormente se le escribe el nombre internamente.

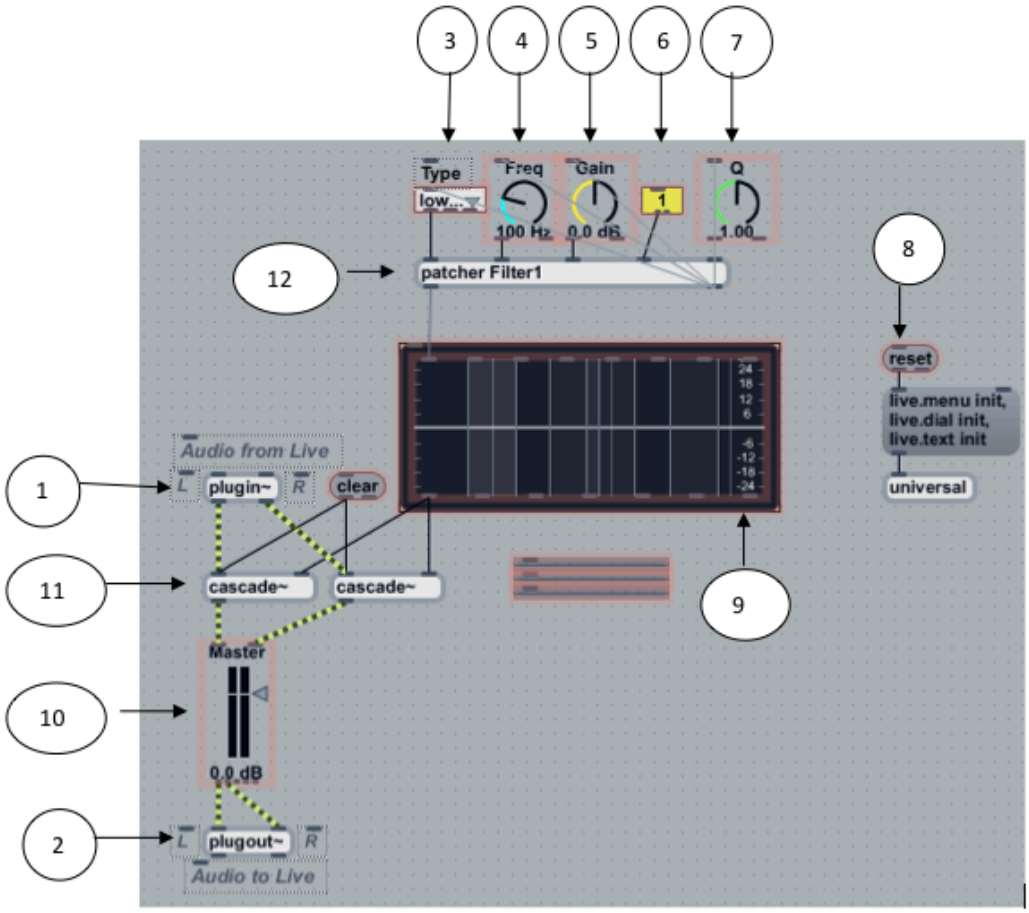
Fig. 15. Ventana de selección de objetos en Max Msp.

El objeto que le sigue a la derecha del general, es de mensaje, también muy utilizado en todo del proceso de creación del proyecto, esta ventana contiene objetos del entorno de *Ableton Live, Max Msp y Jitter.*

El primer paso para la construcción del ecualizador de entrada, será la creación de los parámetros manipulables por el usuario, luego se explica cómo se relaciona con las herramientas de visualización de *Jitter.*

El ecualizador de entrada debe contener todas las características expuestas en el diseño, con los valores de rango y limite para que pueda encajar en la ventana de visualización, estos valores son asignables a cada knob por medio del botón derecho en la ventana de programación y en la pestaña “inspector”, se le pueden dar características de tamaño, valores numéricos, nombre, color etc.… así para cualquier elemento de la ventana de programación.

Fig. 16. Elementos del Algoritmo ecualizador de entrada.



1. Plug-In~: permite recibir la señal de audio para ser manipulada en el entorno de programación de *Max Msp* para *Live*, esta señal llega en estereo, por tanto su salida son dos señales las cuales van de color amarillo con negro, lo que significa que son señales de audio.

2. Plug-Out~: permite retornar la señal, ya procesada en el entorno de programación de Max for live, esta señal llega en estero, y sale igualmente en estéreo a la plataforma de live.

3. Live.menu: Menú desplegable que contiene los diferentes tipos de filtros que se podrán utilizar en el ecualizador, la edición de estos filtros se realiza en el inspector que se activa, haciendo clic derecho sobre el menú desplegable allí se

dan de los filtros que contiene. (*lowpass, highpass, bandpass, bandstop, peaknotch, lowshelf, highshelf, resonant, allpass*)

4. Live.dial "*Freq*": Botón de frecuencia que es manipulado por el usuario, con una variación de rango desde 50 Hz – 18 KHz. Este botón se encuentra en escala exponencial.

5. Live.dial "*Gain*": Botón de Ganancia que es manipulado por el usuario, con una variación de rango desde 0dB – 30 dB. Este elemento le brinda la ganancia a la frecuencia seleccionada desde cero hacia arriba y su escala es lineal.

6. Live.text "1": botón de activación que es manipulado por el usuario para activar o desactivar el filtro.

7. Live.dial "Q": Ancho de banda, con una variación de rango desde 0.1 – 10, este aumenta ó disminuye el ancho de banda del filtro en la frecuencia deseada y su escala es lineal.

8. Live.text "Reset": botón de reinicio, este devuelve todos los valores del ecualizador a los parámetros iniciales, para cuando se desee realizar otra configuración.

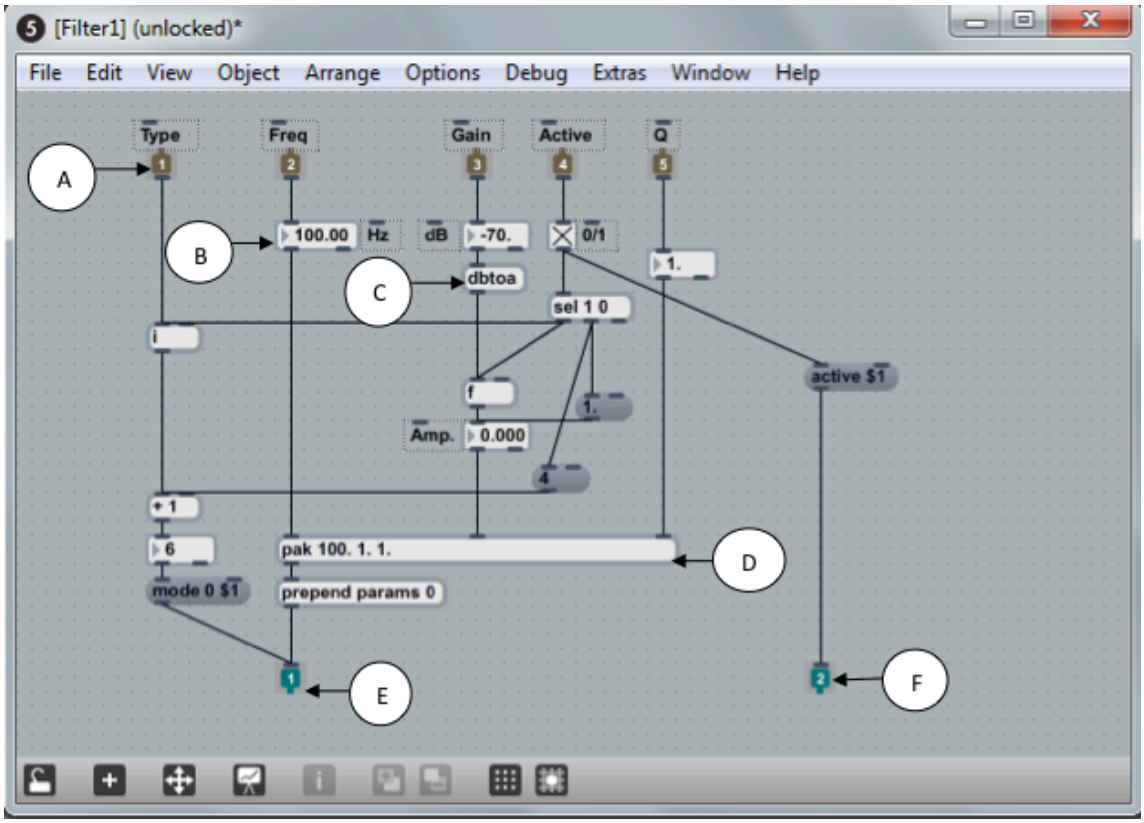
9. Filtergraph~: ventana de visualización del ecualizador, esta permite al usuario observar los cambios que está realizando en función, de la frecuencia contra ganancia.

10. Live.Gain~: permite al usuario adecuar el nivel de salida del ecualizador. Sus valores están dados en decibeles. Con rango de –inf hasta 30 dB.

11. Cascade~: objeto no manipulable por el usuario, es el encargado de aplicar las variaciones del ecualizador de la ventana de visualización del filtro a la señal de audio.

12. Patcher Filter: este tipo de objetos son subprogramas, que contienen en su interior otra programación, estos objetos son creados para organizar mejor la distribución visual del algoritmo. Para acceder a la programación interna de estos subprogramas se hace doble clic sobre él mismo, este contiene el manejo de las variables del filtro manipuladas por el usuario, y su salida va directamente a la ventana de visualización del filtro. A continuación se explicara la construcción interna de este. Al abrirlos se puede observar que el número de salidas y de entradas es igual al del subprograma generado en el programa total, el cual es visualizado en forma resumida.

Fig. 17. Elementos Algoritmo subpatch "Filter1".



A. Inlet "1": este objeto permite la comunicación entre el subpatch y el patch general, es decir por medio de este objeto entra la señal proveniente del menú desplegable para la selección del tipo de filtro, y así respectivamente con los demás inlets.

B. Flonum: permite ver el valor numérico del knob frecuencia que se encuentra en el patch general, este objeto permite visualizar todo tipo de números en formato flotante.

C. Dbtoa: este objeto permite la conversión de decibeles que provienen en escala logarítmica a amplitud en escala lineal. Este rango de conversión estaría dado asi, correspondiendo -70 dB a 0 en amplitud y 30 dB a 31.62 en amplitud respectivamente.

D. Pak: empaqueta los datos provenientes en paralelo desde frecuencia, ganancia y q, a serie, para solo enviar un dato de salida.

E. Outlet "1": este objeto permite la salida de la señal desde el subprograma a la configuración general, por este *Outlet* en particular sale la información que ingreso por todas las entradas del subprograma que van dirigidas luego a la ventana de visualización del ecualizador.

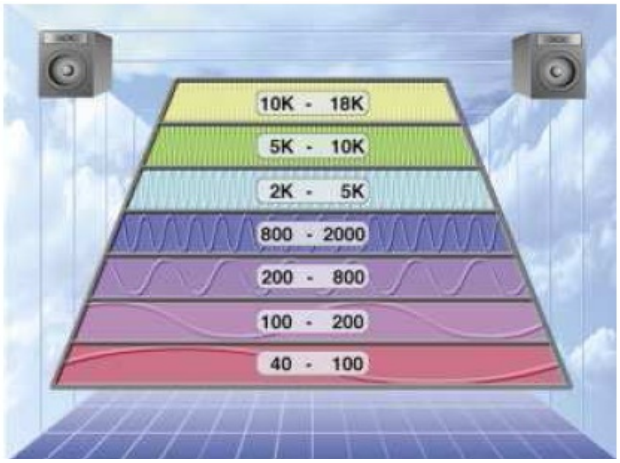
F. Outlet "2": este objeto permite el envió del mensaje de desactivación a todos los objetos ubicados en la interfaz de usuario del ecualizador en *Live.*

De esta manera se completa la construcción del algoritmo ecualizador de entrada, aunque este algoritmo no controla ningún parámetro en la parte visual aún. Finalmente se explicará la relación de estos algoritmos con la visualización tridimensional.

4.3.1. Diseño Del Analizador De Fft.

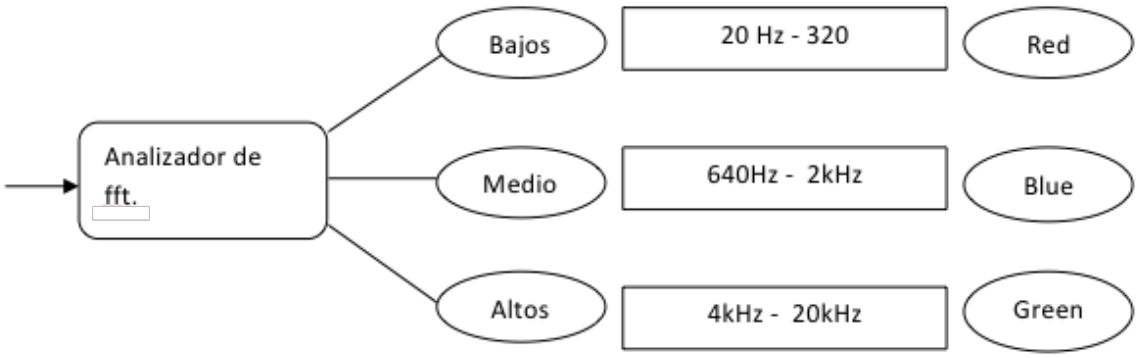
La función de este bloque, es la de recibir en paralelo la señal de audio, proveniente del ecualizador de entrada, y hacerle un análisis en frecuencia, con el fin de caracterizar o diferenciar el sonido de cada canal, es decir, según la teoría de David Gibson, en la visualización se asocian las frecuencias a algunos colores determinados como se muestra a continuación.

Fig. 18. Clasificación de colores según la frecuencia.



Por lo anterior, es necesario un algoritmo que interprete la respuesta en frecuencia del sonido proveniente del canal, para asociarlo a un color definido en los objetos “esferas”. En la ventana de visualización tridimensional, por ejemplo, si en el canal al cual se le ha asignado el *Plug-In* corresponde a un bajo su color debe ser representado a la frecuencia correspondiente entre 100 y 800 Hz, que es el violeta. Además, el ecualizador de entrada, de acuerdo a su configuración ayudará a definir el color y el sonido de cada canal.

Fig. 19. Relación de frecuencias con colores.



Después de analizar las características frecuenciales del canal, se reciben la sumatoria de los valores de amplitud de frecuencias bajas desde 20 Hz a 320 Hz y esta sumatoria promediada es enviada a un objeto que asigna el color a los objetos “esfera”. Si la característica sonora del canal contiene registro en frecuencias bajas se notará de un color rojizo el objeto.

El registro de frecuencias medias se toma desde 640 Hz, hasta 2KHz, haciendo una asignación a color azul, y finalmente las frecuencias altas tomadas desde 4khz a 20 KHz, se les asignara color verde, todo lo anterior aproximándose a las teorías de David Gibson.

Finalmente este bloque de analizador de FFT, no afecta en nada a la señal de audio proveniente del ecualizador de entrada, únicamente analiza sus características frecuenciales, por esta razón, la ubicación de este algoritmo es paralela al flujo de señal.

La interfaz creada en *Live* de este algoritmo debe incorporar un visualizador de FFT y dos controles, uno para modificar la sensibilidad del analizador y otro para controlar el tiempo de toma de los datos.

Fig. 20. Diseño del algoritmo analizador fft en Live.

1. Ventana de visualización de FFT

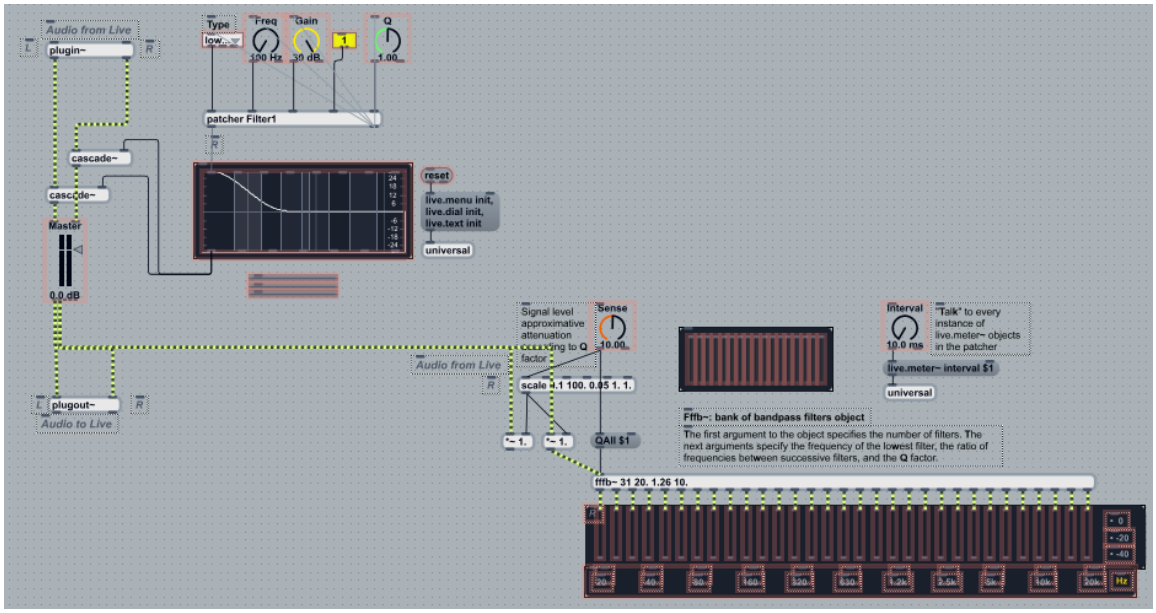
2. Botón de ajuste de sensibilidad, rango (1 - 100)

3. Botón de intervalo de recepción de muestras, rango (10ms – 1s)

4.3.2. Construccion Del Analizador De Fft.

En la construcción de este algoritmo se toma la señal de audio en forma paralela, esta es representada por las líneas intermitentes amarillo con negro. El ensamble completo hasta ahora contiene entonces el ecualizador de entrada y el analizador de FFT como se ve en la figura 21.

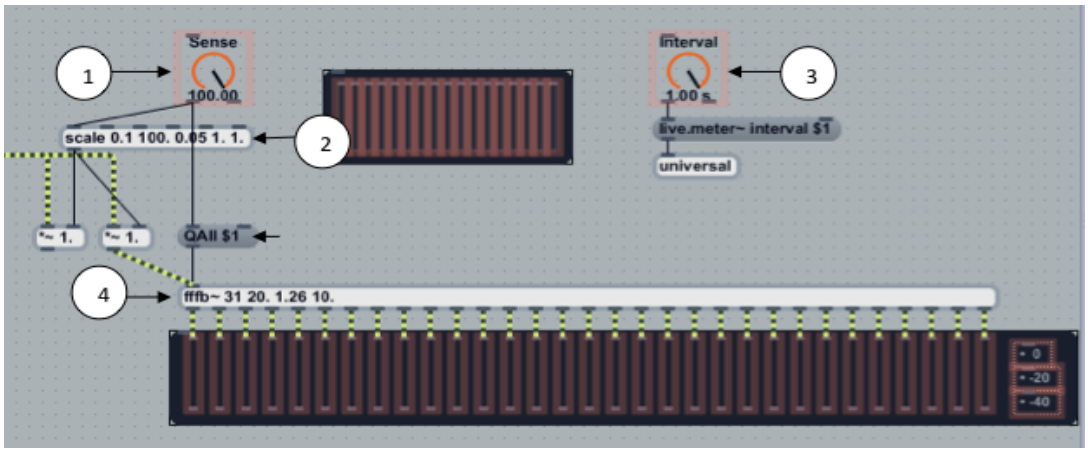
Fig. 21. Construcción del algoritmo analizador fft en Max Msp visión general.



4.3.3 Objetos Utilizados En El Algoritmo Analizador De Fft.

A continuación se muestran los objetos utilizados en la construcción del algoritmo analizador de FFT, con su correspondiente explicación y función dentro del ensamble.

Fig. 22. Elementos del algoritmo analizador fft



1. Live.dial "sense": control de sensibilidad, permite al usuario variar la sensibilidad en amplitud del analizador de frecuencia, su rango varía desde 1 a 100, en escala lineal, cuando este valor es menor, se aproxima la respuesta al Q del ecualizador, cuando este valor es mayor, aumenta la amplitud en la mayoría de frecuencias cercanas a la fundamental.

2. Scale: este objeto, muy importante en la realización de muchos algoritmos permite, escalar la variación de parámetros a valores deseados, por ejemplo en este caso, es necesario pasar el rango de valores de knob de sensibilidad, desde 1 – 100 a 0.5 – 1.

3. Live.dial "interval": control de intervalo, este parámetro permite al usuario variar el intervalo de toma de muestras de la señal de audio, su rango varía desde 10ms a 1s, la función de este, es estabilizar las muestras cuando persistan muchas variaciones en frecuencia.

4. FFFB~: *fast fixed filter bank*, este objeto está compuesto por una serie de filtros los cuales son asignados a cada banda de frecuencias, este filtro tiene el número de salidas definidas por el usuario.

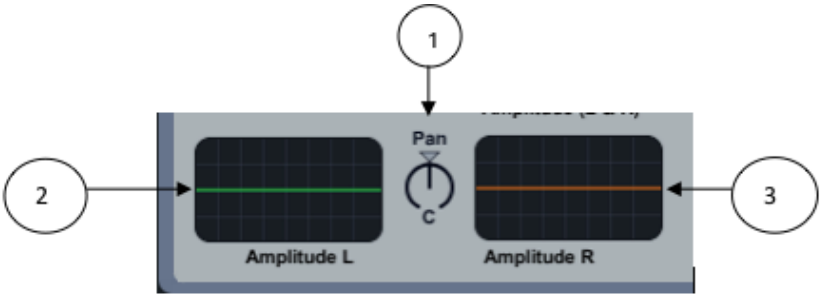
De esta manera se finaliza la construcción del algoritmo analizador FFT, la relación de este algoritmo con los objetos de visualización tridimensional se explicaran en la parte del diseño con *Jitter.*

4.4.1 **Diseño Del Algoritmo Paneo**.

La función de este bloque es tomar la señal de audio desde el algoritmo ecualizador de entrada y balancear la señal a la izquierda ó derecha, dependiendo del criterio del usuario, con el fin de espacializar horizontalmente el canal, en la ventana de visualización tridimensional.

La interfaz de usuario creada en *Live* deberá contener un control que permita la variación de paneo, así como dos ventanas visualizadoras de la señal que se está manipulando de derecha e izquierda.

Fig. 23. Diseño del algoritmo paneo en Live.



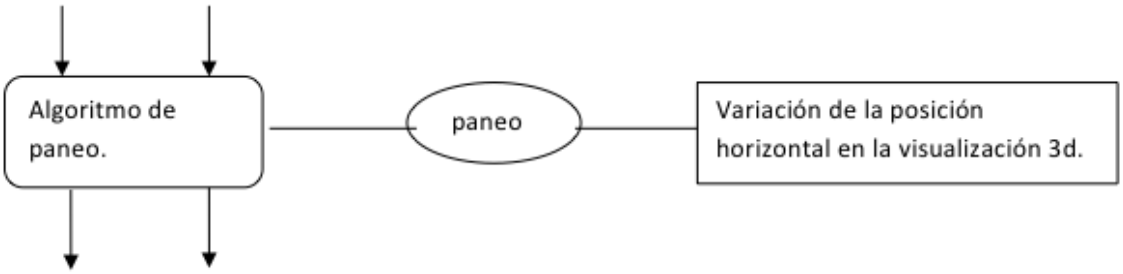
1. Control de paneo, rango (-50 a 50)

2. Ventana de visualización de amplitud L.

3. Ventana de visualización de amplitud R.

El rango de variación del paneo va desde -50 que corresponde a que toda la señal de audio este balanceada hacia la izquierda, y 50 cuando la señal de audio se encuentra balanceada hacia la derecha.

Fig. 24. Diagrama flujo Paneo.



Los parámetros enviados desde el botón de paneo deben ser enviados a los objetos de visualización de *Jitter*, los cuales desplazaran el objeto “esfera” hacia la izquierda o la derecha dependiendo de la selección del usuario.

Este algoritmo va en serie con el del ecualizador de entrada, por lo tanto, la salida del algoritmo paneo contendrá los cambios de señal realizados previamente para la siguiente etapa.

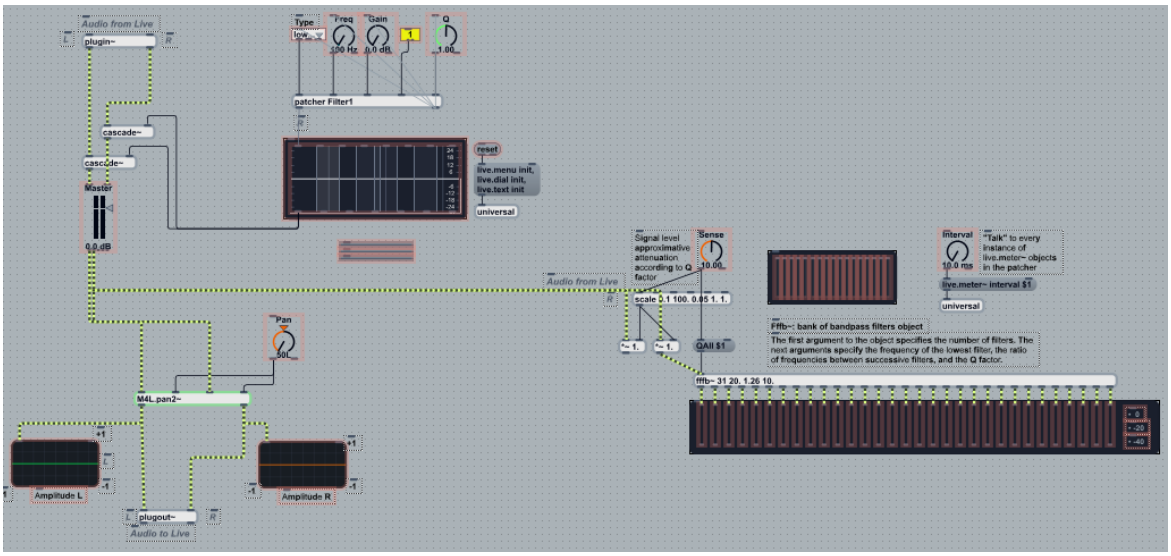
4.4.2. Construccion Del Algoritmo Paneo.

En la construcción de este algoritmo se toma la señal de audio proveniente del bloque anterior y se adecua de acuerdo a los siguiente objetos explicados a continuación en su funcionamiento y clase.

Las opciones de visualización tridimensional de este algoritmo influyen dependiendo los valores numericos que el usuario registre en el movimiento del knob pan, en el plugin, estos valores cambian unos parametros de posicion de las opciones de Jitter y por consecuencia se obserba un movimiento del objeto “esfera ” en la ventana de visualización.

En la parte superior de la figura 25, se observa el algoritmo ecualizador, en la parte derecha el algoritmo analizador de FFT y siguiendo la secuencia lógica de la programación en la parte inferior el algoritmo paneo.

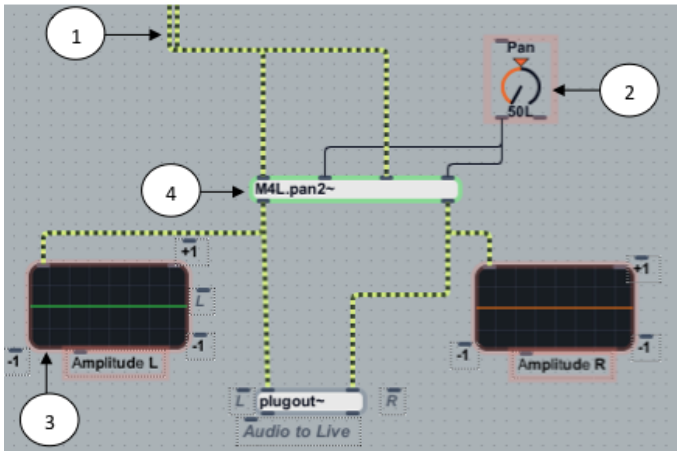
Fig. 25. Construcción del algoritmo paneo en Max Msp visión general.



4.4.3 Objetos Utilizados En La Construccion Del Algoritmo Paneo.

A continuación se muestran y explican los objetos utilizados en la construcción del algoritmo paneo.

Fig. 26. Elementos del algoritmo paneo.



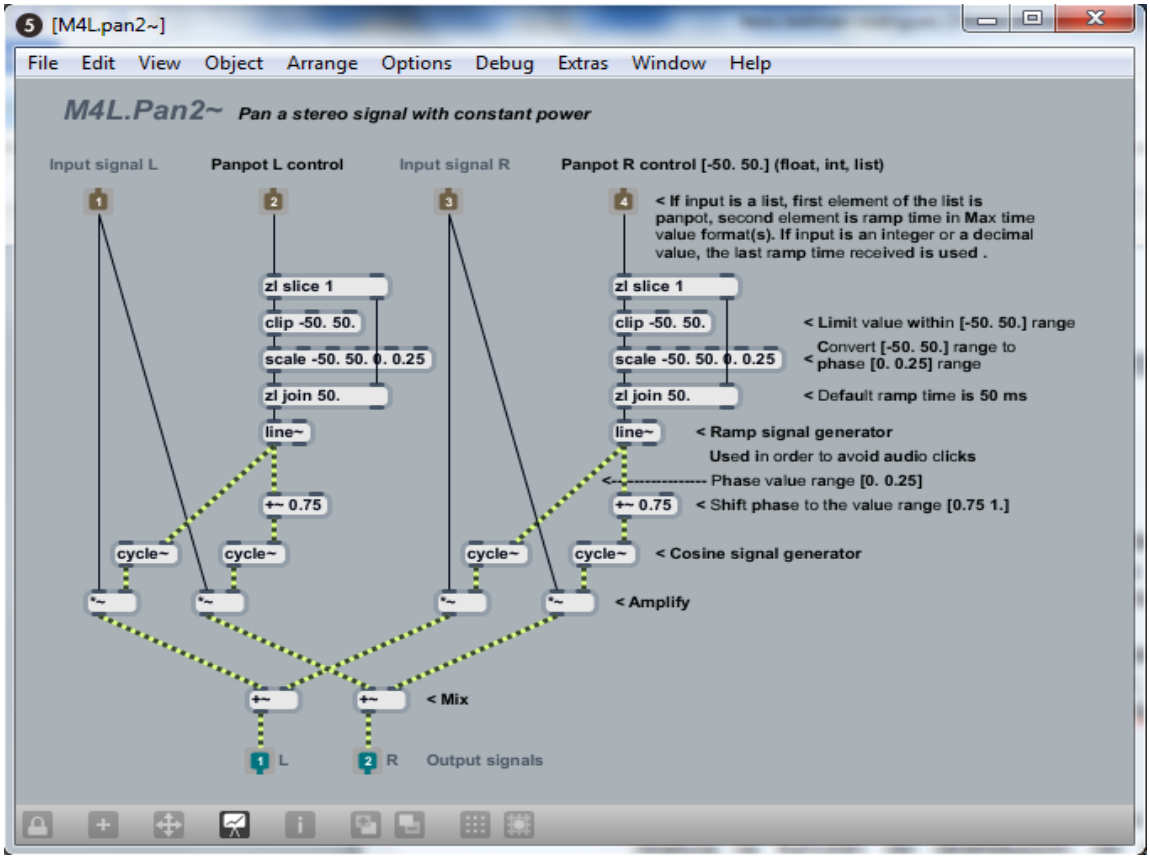
1. Señal de audio proveniente del anterior bloque, “algoritmo ecualizador de entrada”

2. Live.dial “Pan”: control de paneo, este objeto permite al usuario seleccionar la cantidad de señal de audio que se va a desplazar a la derecha o la izquierda, su rango es de – 50 completamente a la izquierda, y 50 completamente a la derecha, su valor por defecto se encuentra en 0 es decir en el centro.

3. Scope~ “Amplitude L”: esta es la ventana de visualización para la amplitud de la señal L, permite ver el nivel de señal, mientras el botón paneo se encuentre hacia la izquierda, igualmente la ventana Amplitude R, cumple la misma función, pero al lado derecho.

4. Patcher “M4L.Pan2~”: este objeto es un subprograma, el cual realiza la función de distribución de señal hacia la derecha o izquierda, dependiendo de los parámetros de entrada definidos por el usuario, y su salida conlleva la señal de audio ya procesada hacia el entorno de *Live.*

Fig. 27. Elementos Algoritmo subpatch “M4L.Pan2”.



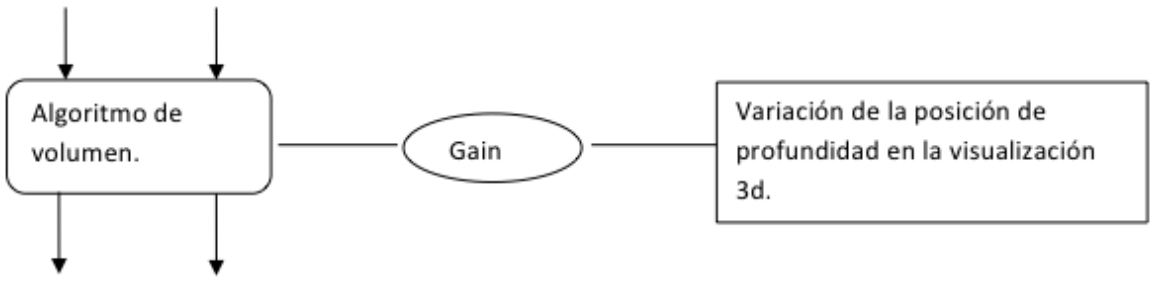
La función de este algoritmo es tomar la señal estéreo proveniente a fin de separarla para ser procesada independientemente por cada lado, el control de paneo varia en las entradas *Panpot* L y R control, generando un cambio de fase en la onda para poder realizar por medio de interferencia destructiva la cancelación de uno de los lados de la señal estero, ya sea L ó R.

4.5.1 Diseño Del Algoritmo Gain "Volumen".

Esta es la última parte del proceso de audio, en este bloque se toma la señal proveniente de todos los algoritmos anteriores y su función es dar mayor nivel a la señal finalmente para la salida del *Plug-In*.

Este algoritmo al variar el nivel de amplitud de la señal, tendrá una variación igualmente en la ventana de visualización, característica determinada por la profundidad del objeto en el espacio tridimensional, es decir mientras más alejado este el objeto de la parte frontal, menor nivel de amplitud tendrá, y mientras más cercano este de la parte frontal será mayor su nivel de amplitud.

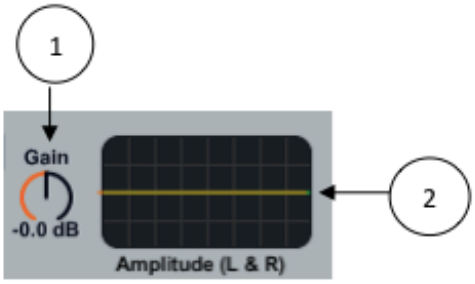
Fig. 28. Diagrama de flujo algoritmo Volumen.



Este algoritmo va en serie con los algoritmos de las etapas anteriores por lo tanto la salida del algoritmo paneo contendrá los cambios de señal realizados previamente para la salida de vuelta al entorno de *Live.*

La interfaz de usuario creada en *Live* deberá contener un control que permita la variación de volumen, así como una ventana para visualizar el cambio de la señal.

Fig. 29. Diseño del algoritmo volumen en Live.



1. Control de Ganancia, rango (-inf a 30dB)

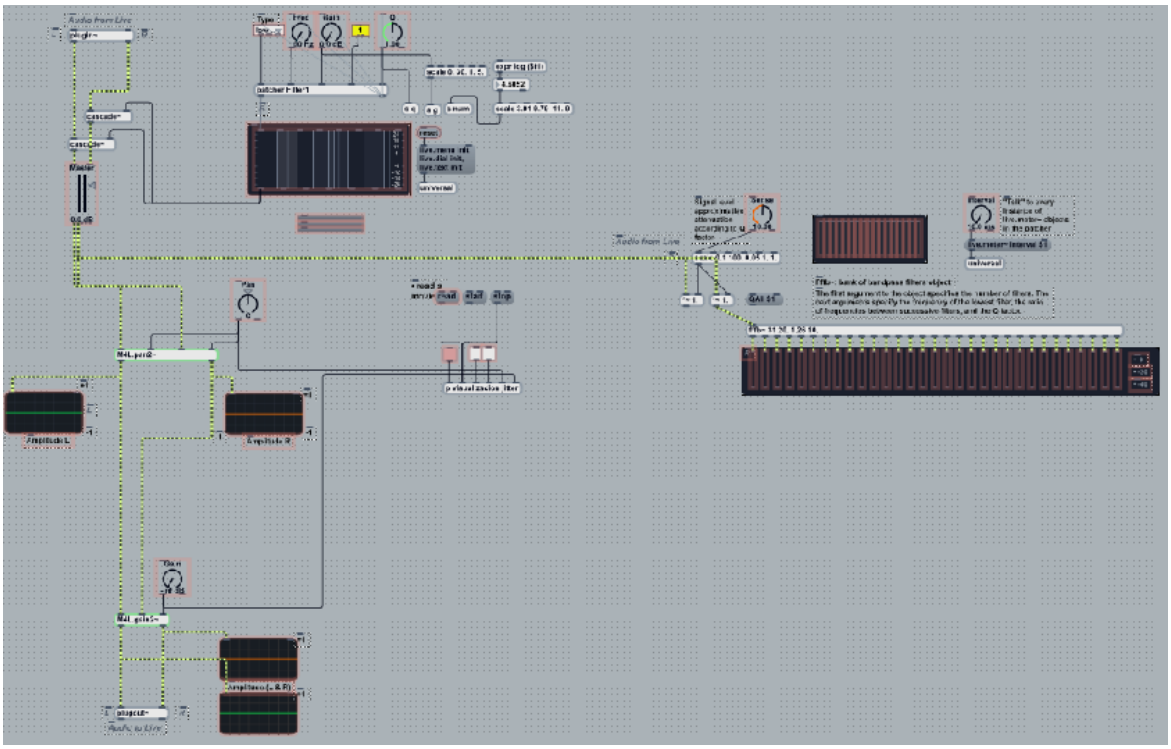
2. Ventana de visualización de nivel de amplitud L y R.

El rango de variación del control de ganancia va desde -inf que corresponde a que toda la señal de audio este en nivel nulo, y 30dB cuando la señal de audio se encuentra en el nivel máximo de amplitud.

4.5.2 Construccion Del Algoritmo Volumen.

En la construcción de este algoritmo se toma la señal de audio proveniente del bloque anterior y se adecua de acuerdo a los siguiente objetos explicados a continuación en su funcionamiento y clase.

Fig. 30. Construcción del algoritmo paneo en Max Msp visión general.

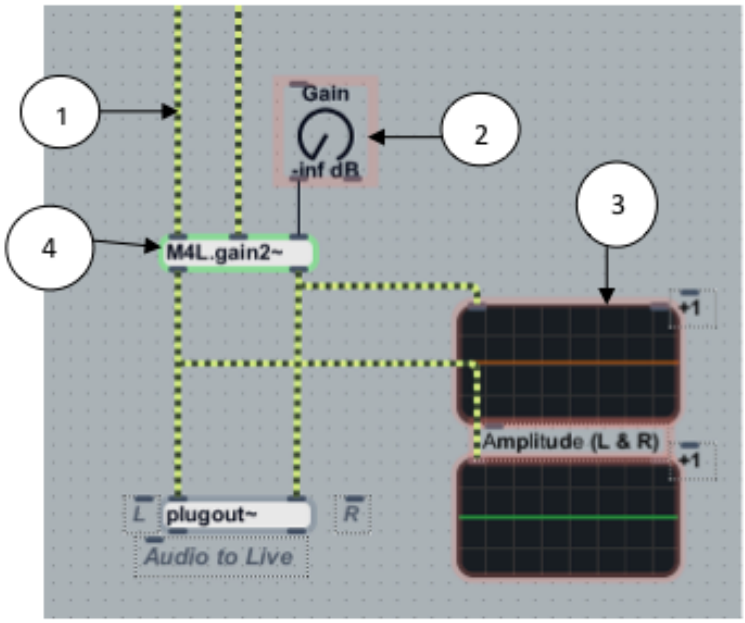


Este es el esquema general de toda la programación que se ha realizado hasta el momento, en la parte superior se encuentra el algoritmo ecualizador, en la parte derecha el algoritmo analizador de FFT, en la parte central el algoritmo de paneo y finalmente el algoritmo para el control de volumen en la parte inferior.

4.5.3 Objetos Utilizados En La Construccion Del Algoritmo Volumen.

A continuación se muestran y explican los objetos utilizados en la construcción del algoritmo volumen.

Fig. 31. Elementos del algoritmo volumen.



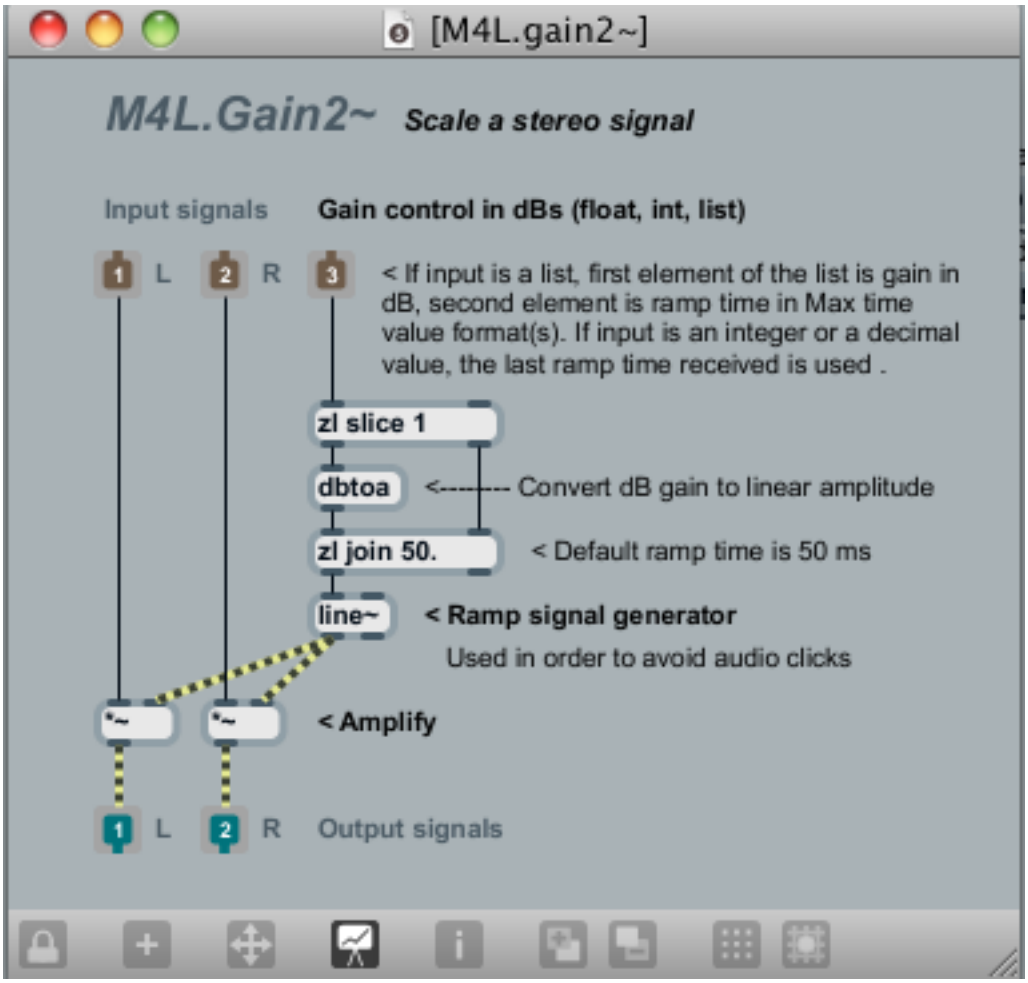
1. Señal de audio proveniente del anterior bloque, “algoritmo paneo”.

2. Live.dial “Gain”: control de volumen, este objeto permite al usuario seleccionar la cantidad de señal de audio que se va a amplificar, su rango es de – inf dB completamente a la izquierda, y 30 dB completamente a la derecha, su valor por defecto se encuentra en 0 es decir en el centro.

3. Scope~ “Amplitude L & R”: esta es la ventana de visualización para la amplitud de la señal L y R, permite ver el nivel de señal en tiempo real.

4. Patcher “M4L.Gain2~”: este objeto es un subprograma, el cual realiza la función de recibir las variaciones del knob Gain y convertir estos valores de dB a escala lineal, y su salida conlleva la señal de audio ya procesada hacia el entorno de *Live.*

Fig. 32. Elementos Algoritmo subpatch "M4L.Gain2".



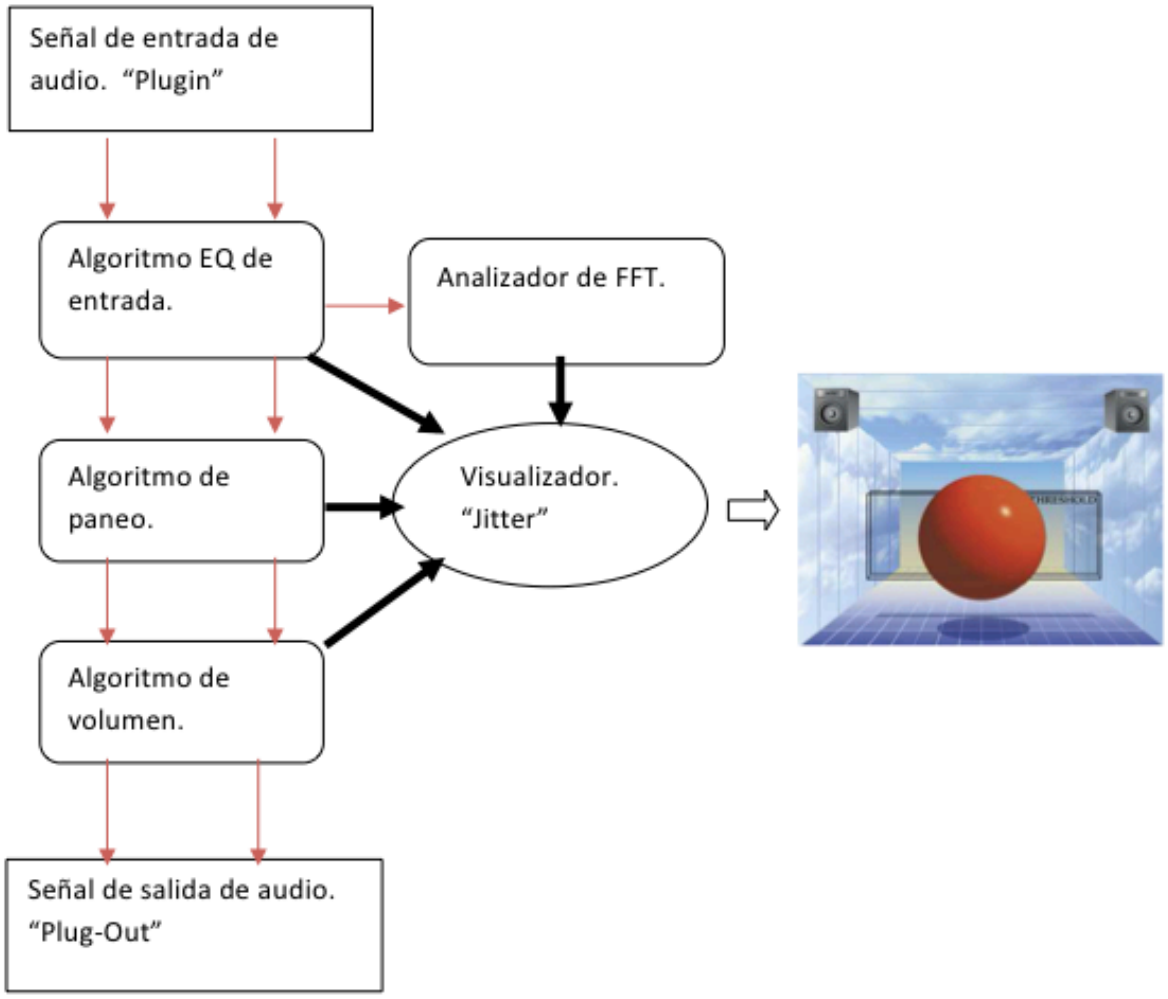
Este algoritmo realiza la amplificación de la señal, la conversión final de dB a escala lineal y la salida final de este algoritmo sera la salida final de todo el algoritmo general.

4.6.1 Visualizador Jitter.

Jitter es la extensión de Max Msp, desarrollada en el 2003, esta herramienta permite procesar video, 3D y matrices en el programa.

El algoritmo general aparte de contener los algoritmos de proceso de audio es habido de utilizar algunos objetos de jitter, el visualizador tomara secciones del ecualizador, el paneo, fft, y volumen para lograr la interpretación tridimensional, en una ventana externa que se abrirá automáticamente al cargar el plug-in en ableton live 8.

Fig. 33. Diagrama de flujo algoritmo General.



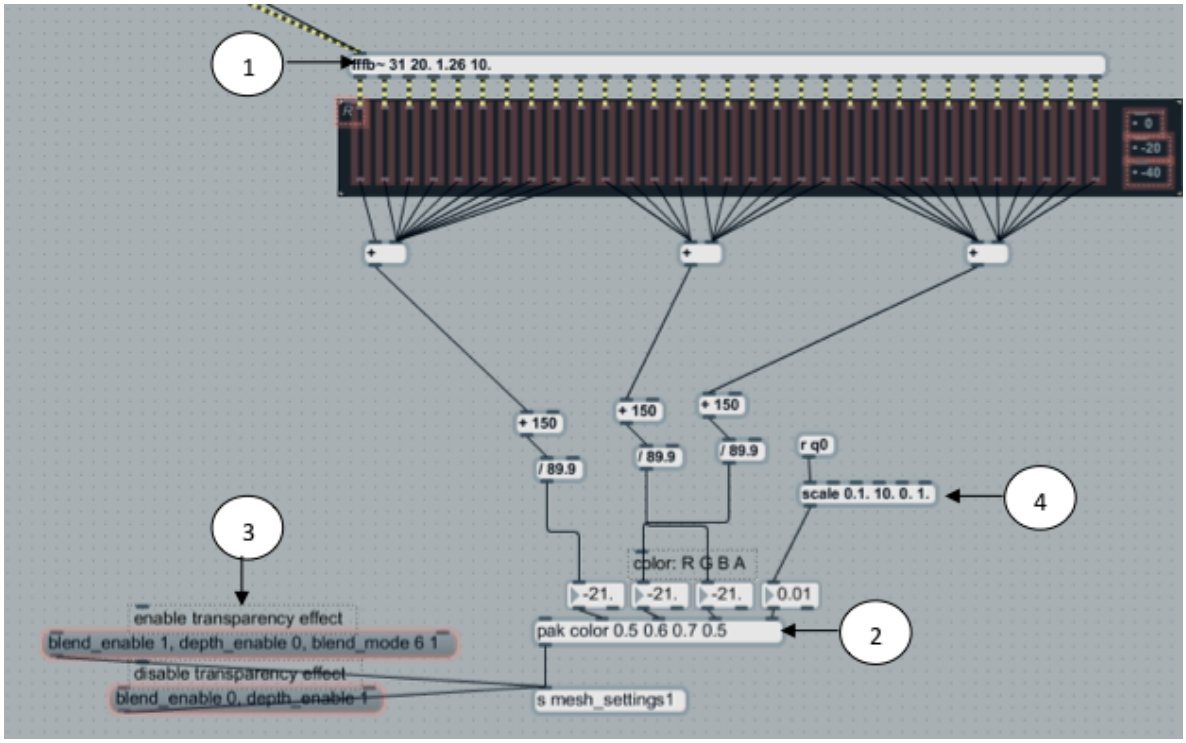
El Visualizador jitter en la ventana de visualización, carga una esfera por canal, sobre un fondo negro, es necesario cargar un fondo en formato .JPEG esta esfera tomara variables del analizador fft para obtener un color dependiendo del sonido, del algoritmo paneo para su ubicación horizontal, del algoritmo EQ de entrada

para su ubicación vertical, y del algoritmo de volumen para su ubicación de profundidad.

4.6.2 Propiedades De Color Del Visualizador Jitter.

La característica del color que se va a asignar al objeto “esfera”, se toma del analizador fft. El algoritmo analizador fft, contiene un detector de frecuencia de 30 bandas, el objeto que manipula el color en jitter “pak color” maneja un formato de color RGB (Red Green Blue), para controlar estos colores con las 30 variables provenientes de las bandas del analizador es necesario agruparlas por sumatorias y promediarlas para que únicamente queden tres variables que correspondan a los colores: Rojo, Verde y azul. Es decir tomar frecuencias bajas para el rojo, medias para el azul y altas para el verde, estas convenciones concuerdan con la teoría de color de David Gibson.

Fig. 34. Parámetros de color desde la fft.



El objeto “1” explicado anteriormente es el analizador de fft, las sumatorias se toman en tres grupos de 10 bandas de frecuencia cada uno, es necesario sumarle

la constante 150 y posteriormente dividir este valor en 89.9 para promediar la sumatoria y que estos valores estén en el rango para la caracterización del color.

El objeto “2” toma los valores numéricos de los tres grupos de frecuencias y un cuarto valor proveniente del ecualizador de entrada, los tres valores frecuenciales afectan los colores de la esfera mientras que el cuarto valor afecta la intensidad del color es decir la transparencia del objeto, este valor proviene del factor Q del ecualizador de entrada, cuando mayor es el valor Q, es decir el ancho de banda mas restrictivo, la intensidad del color es mayor, y cuando el valor Q es menor, el objeto será mas translucido.

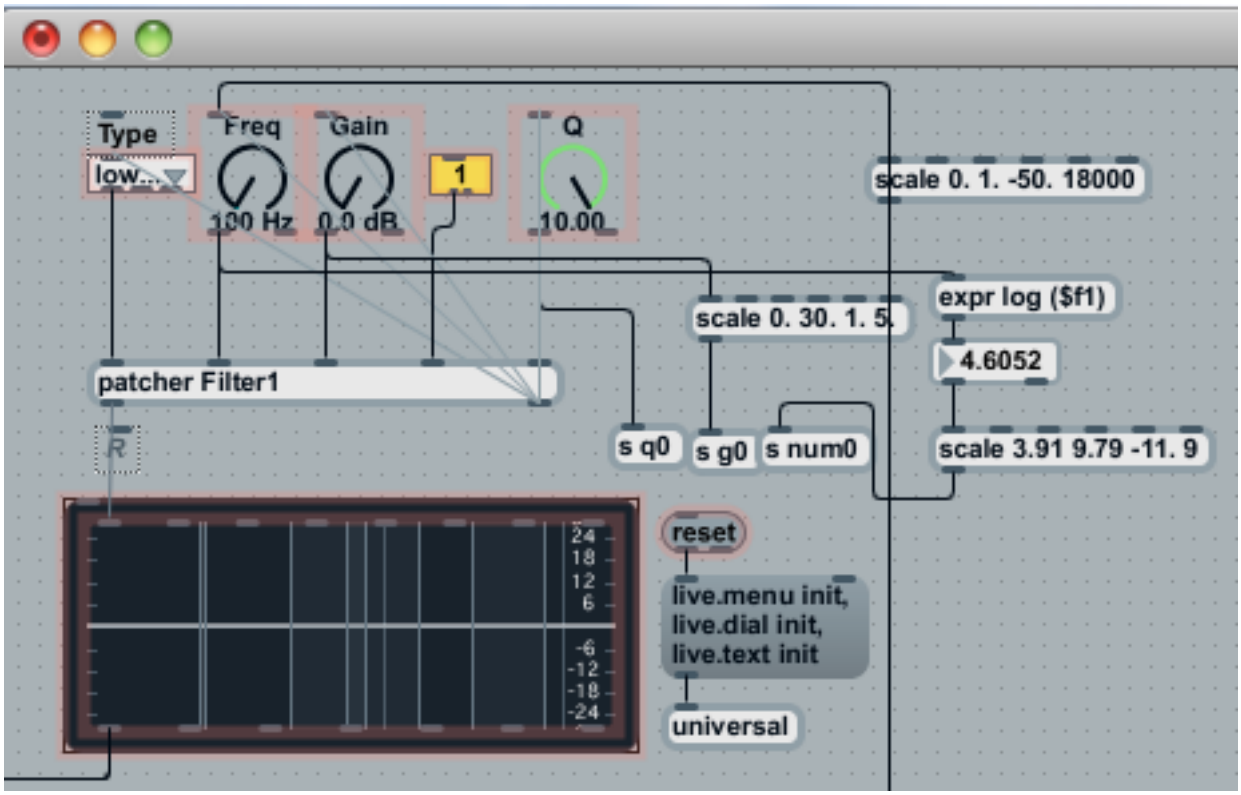
Los objetos “3” son visibles en el plug-in, sobre el entorno de ableton live, estos objetos permiten la activación y la desactivación de la transparencia de los objetos, el objeto superior “blend_enable 1” activa la transparencia, mientras que el objeto “blend_enable 0” desactiva la transparencia de los objetos, por defecto, los objetos se cargan sin transparencia.

El objeto “4” permite cambiar la escala proveniente del factor Q del ecualizador de entrada del rango entre (0.10 - 10) a los valores que activan la transparencia entre (0 y 1).

4.6.3 Propiedades Dimensionales Del Visualizador Jitter.

Los objetos cargados en la ventana de visualización son esferas, por tal razón las dimensiones únicamente serán variadas por una variable que manipule el diámetro, la variable encargada de esta función es la ganancia del filtro que proviene del ecualizador de entrada.

Fig. 35. Relación algoritmo ecualizador de entrada con propiedades dimensionales.



Esta variable se encuentra en el rango comprendido entre (0dB y 30 dB) es necesario escalar estos valores a valores permisibles para la visualización máxima radial de la esfera, (1 para el radio mínimo de la esfera y 5 para el valor máximo radial).

4.6.4 Propiedades De Posición Del Visualizador Jitter.

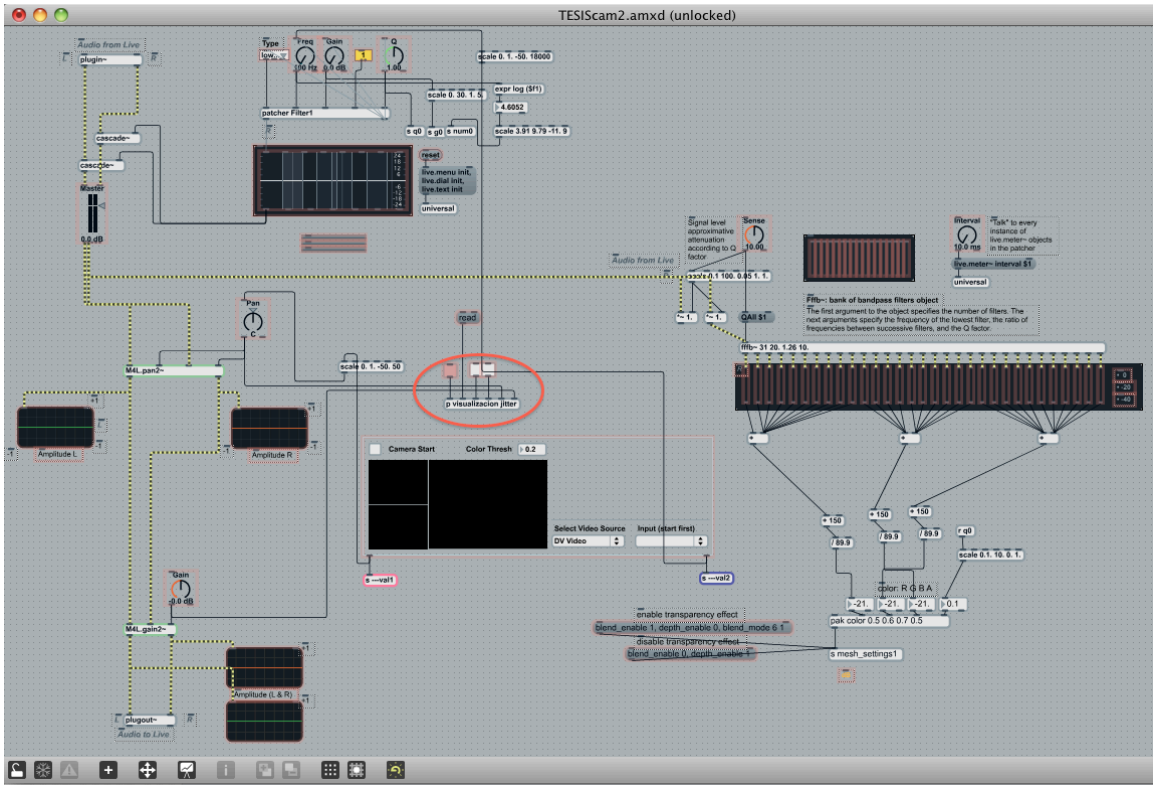
Los parámetros de posición son: profundidad delimitada por el volumen, altura controlada por la frecuencia, posición horizontal controlada por el paneo.

Los limites de los valores espaciales están ubicados en la figura 14, estos valores son obtenidos de la visualización de *jitter.* Los valores para la variación horizontal “paneo” se encuentran en escala lineal en el rango comprendido entre (-9 y 9), los valores verticales “frecuencia” están en escala lineal, por esta razón es necesario realizar una conversión a escala lineal para ubicarlos en el rango de visualización comprendido entre (-9 y 11), los valores de profundidad “volumen” permanecen en el rango entre (-67 y -30).

4.6.5 Sub Patch Visualización Jitter.

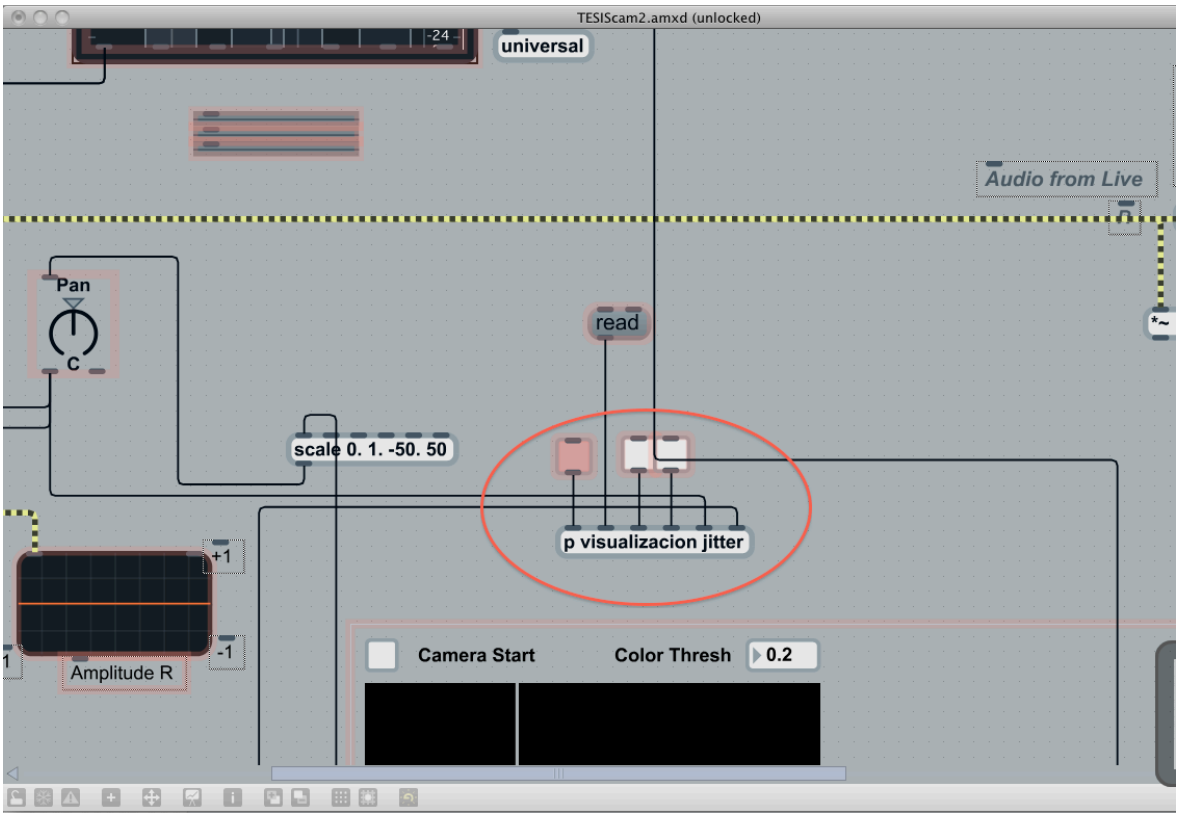
La mayoría de las funciones del visualizador como, la forma del objeto a visualizar, la posición y el tamaño son programadas al interior de este subpatch, que se encuentra en la parte central del algoritmo general. A continuación se explicaran detalladamente los elementos que componen este subpatch en su interior.

Fig. 36. Subpatch visualización jitter, en algoritmo general.



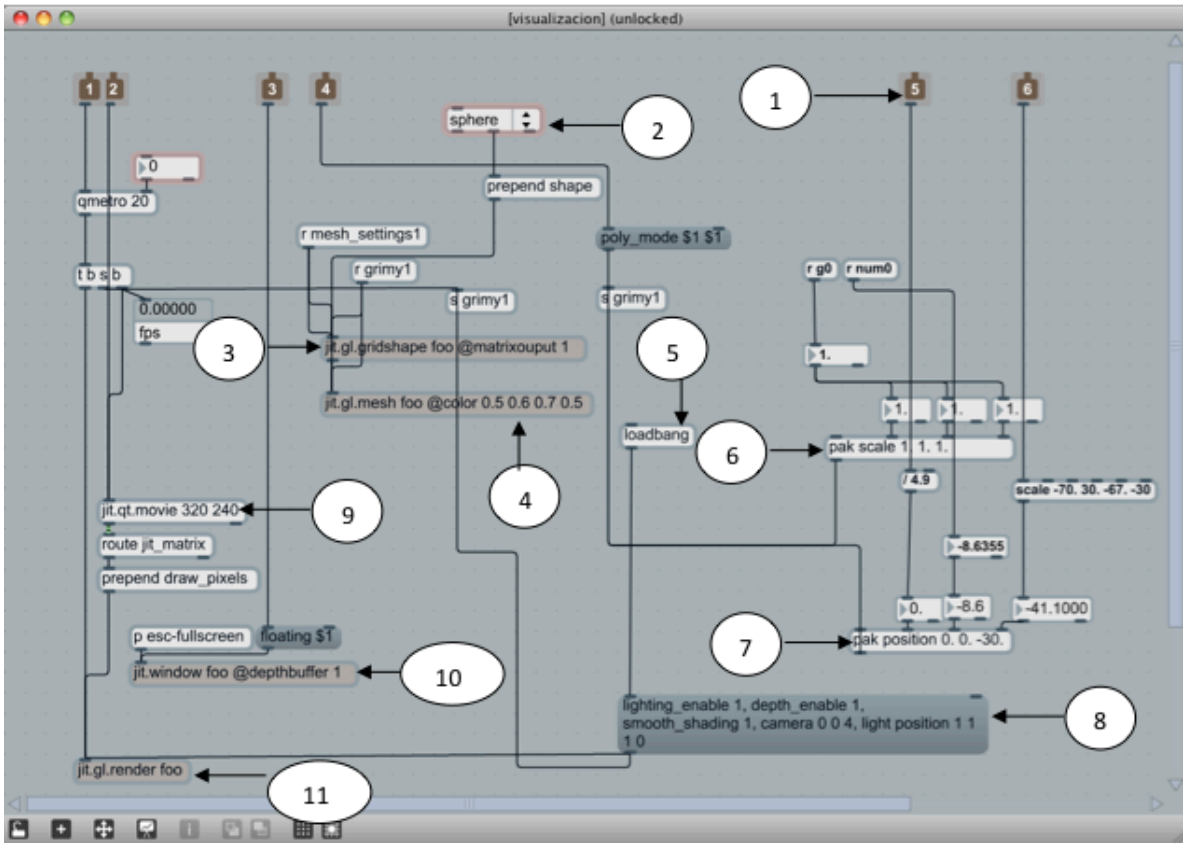
Las 6 entradas a este subpatch son elementos que controlan partes de la ventana de visualización. Algunas variables se envían por elementos “send” y en la parte interior del subpatch visualización se reciben con elementos “receive” para simplificar la distribución y claridad en el algoritmo general además de disminuir el numero de entradas al subpatch visualización.

Fig. 37. Entradas al subpatch visualización jitter.



para entrar a la programación de este subpatch, únicamente se le oprime doble click en el objeto que contiene toda la información de este subprograma llamado “p visualización jitter”, automáticamente se abrirá en una nueva ventana la programación de este subpatch.

Fig. 38. Elementos Algoritmo Subpatch visualización Jitter.



Inlets "1, 2, 3, 4, 5": permiten la entrada de variables y activaciones al subpatch visualización. De izquierda a derecha están: "1" switch de activación para la ventana de visualización, "2" botón de carga de imagen de fondo para la ventana de visualización, "3" Visualización en modo vértice, "4" bloqueo del tamaño de la ventana de visualización, "5" entrada de la variable de paneo, "6" entrada de la variable volumen.

Umenu: menú desplegable para seleccionar el tipo de objeto a visualizar, entre los objetos disponibles se encuentran: esfera, toroide, cilindro, cilindro abierto, cubo, cubo abierto, plano y circulo.

Jit.gl.gridshape: objeto de la extensión jitter, este objeto permite graficar formas tridimensionales o bidimensionales en la ventana de visualización y almacenarlos

en una matriz para ser manipulados de múltiples maneras, como operaciones con otros objetos como, suma, diferencia, multiplicación, derivación…este objeto requiere un argumento de jit.window, “foo” que es el nombre de la ventana donde se van a visualizar los objetos.

Jit.gl.mesh: permite referenciar todos los objetos con un plano de coordenadas tridimensionales, debe contener como argumento igualmente el nombre de la ventana jit.window “foo”, contiene además otro argumento color, que permite visualizar en tres colores el plano de coordenadas cuando este se active, de color verde, azul y rojo.

Loadbang: envía un dato de activación a todo el subpatch cuando el algoritmo es abierto, es decir activa automáticamente el patch al cargar el programa.

Pak scale: permite la variación del tamaño del objeto, por defecto esfera, es decir permite la variación del radio de la esfera, los valores que controlan este objeto provienen del objeto “r g0” (receive go) que proviene del Gain del ecualizador de entrada por medio del objeto “s g0” (send g0)

Pak position: controla la ubicación en las tres variables dimensionales x,y,z, por la entrada 5 controla la primera variable “x horizontal” (paneo), la segunda variable “y vertical” (r num0) proviene desde el envío (s num0) desde el controlador de frecuencia del ecualizador de entrada, y la tercera variable “z profundidad” proviene por la entrada 6 que contiene el control de volumen.

Message: en este objeto se dan las condiciones iniciales de visualización, propiedades del renderizado de las imágenes, por ejemplo: luz, profundidad, posición de la cámara (punto de vista hacia los objetos), posición de la luz.

Jit.qt.movie: permite cargar una imagen o un video en formato .mov de fondo en la ventana de visualizacion, es necesario tener instalado *QuickTime* para que la visualización sea posible, esta imagen o video es almacenada en forma matricial para ser operada junto con los objetos visuales creados en el algoritmo.

Jit.window: permite abrir una ventana externa en donde visualizar todos los objetos el nombre de la ventana se ubica en este objeto, “foo” el argumento depthbuffer permite que los objetos puedan ubicados en partes profundas puedan ser vistos por medio de transparencias ante objetos ubicados en partes cercanas.

Jit.gl.render: permite el renderizado de los objetos creados en el algoritmo, por medio de las extensiones OpenGl de jitter. El argumento que acompaña esta función es el nombre de la ventana jit.window “foo”.

4.7.1 Desarrollo De Alcances.

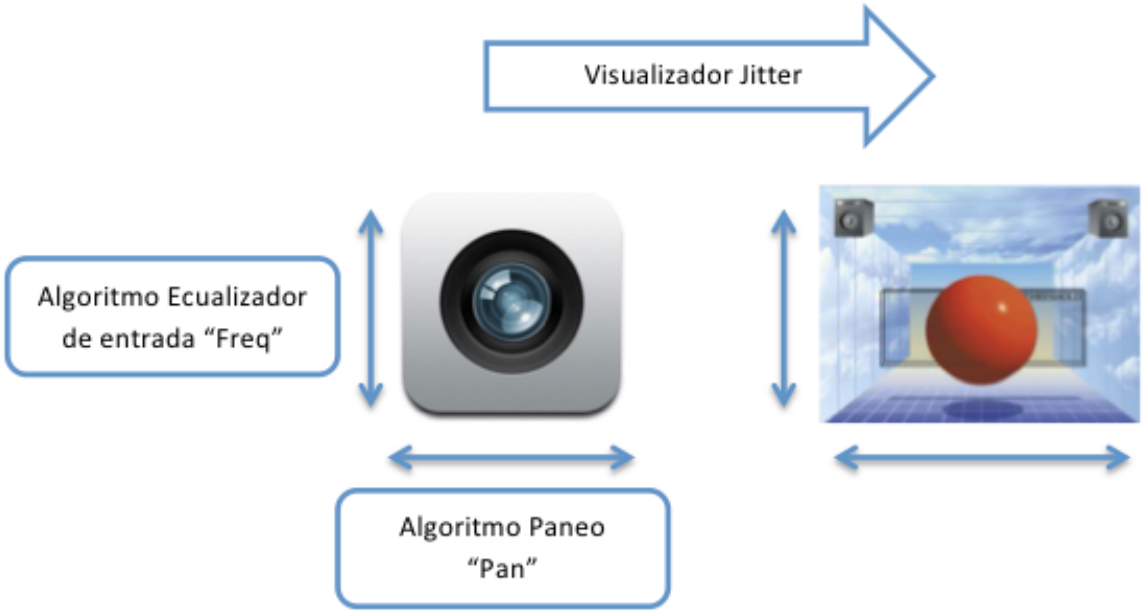
Dado el desarrollo del proyecto fue viable la realización de uno de los alcances propuestos en el anteproyecto, este avance es la programación adicional de un algoritmo que permita la interacción con el plug-in por medio de una cámara.

La cámara será en este caso la encargada de manipular dos variables, posición horizontal “paneo” y vertical “frecuencia”, por tal razón será un manejo netamente bidimensional. Este algoritmo permite seleccionar el color con el cual se va a controlar el objeto “esfera” en la ventana de visualización, es decir, mientras la cámara este activada, una ventana en el plug-in permite seleccionar que color de la imagen que se esta registrando será la que cambie los valores en el algoritmo. En el anexo 4. Se muestra un ejemplo del funcionamiento de este algoritmo que intenta explicar de una manera mas clara la forma de interacción del algoritmo cámara incluido en el plugin.

El siguiente diagrama de bloques explica la relación entre las dimensiones que captura la cámara, que son vertical y horizontal. La cámara registra el movimiento horizontal y vertical, este movimiento a través del algoritmo visualizador jitter, transforma las variables frecuencia del algoritmo ecualizador de entrada, y paneo, estas variables cambiaran la posición del objeto “esfera” en la ventana de visualización.

Es necesario permanecer en un ambiente donde no haya mucho contraste entre colores para que la cámara pueda reconocer mas fácilmente el movimiento e interpretarlo en el algoritmo.

Fig. 39. Diagrama de relación entre la cámara y la ventana de visualización.



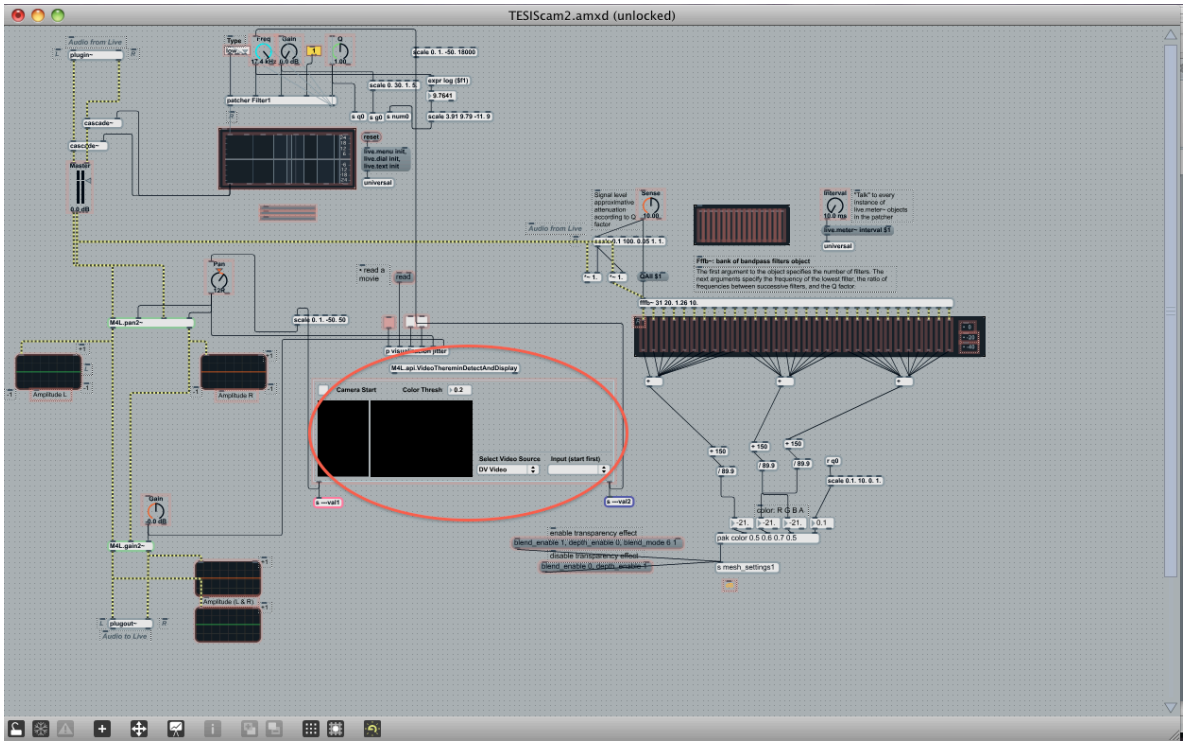
En este caso se utilizo la cámara integrada del computador para la captura de movimiento, es posible utilizar una cámara externa ya que el algoritmo permite seleccionar la entrada de video de una cámara distinta a la integrada, esta opción permite otro avance si se desea tener una interacción completamente tridimensional, programando la variación de volumen desde la captura de movimiento.

4.7.2 Construccion Del Algoritmo De Interaccion Por Camara.

Para el ensamble de esta sección se uso un algoritmo de la librería incluida en max for live, "M4L.api.VideoThereminDetectAndDisplay" este algoritmo activa la entrada desde la cámara, permite seleccionar en la pantalla donde se observa el video, un color el cual va a ser el punto de referencia para el movimiento y por medio de sus salidas obtener estos movimientos como cambios numéricos, estos cambios numéricos van directamente a controlar el knob "Freq" en el ecualizador de entrada y el knob "Pan" en el algoritmo paneo, de esta manera ya se tiene control de estos parámetros, si se deseara manejar en otras variables, únicamente

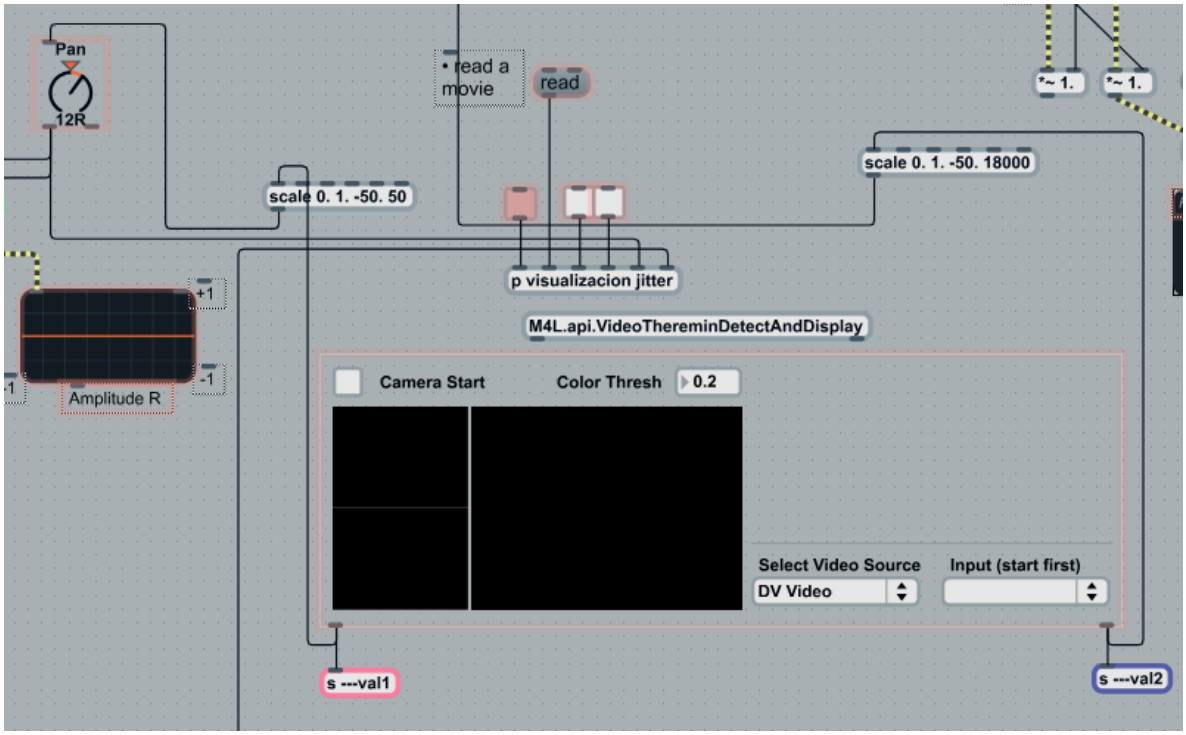
es necesario asignar las salidas del subpatch “M4L.api.VideoThereminDetectAndDisplay” a las variables a controlar.

Fig. 40. Receptor movimiento por cámara en algoritmo general.



Los limites del cuadro de captura de la cámara están de (0 a 1) tanto horizontal como vertical, por esta razón es necesario escalar este rango a los datos de variación de los knobs, paneo y freq. La variación de paneo va desde (-50 a 50) y el valor de frecuencia desde (-50 a 18000) para que tomen todos los valores horizontales y verticales. Esta variación de escala se realiza mediante la función “scale” los dos primeros valores son los valores limite de la ventana, y los siguientes dos valores son a los que se desean las variaciones.

Fig. 41. Algoritmo Receptor por cámara, "M4L.api.VideoThereminDetectAndDisplay"



Para cargar este algoritmo en Max, se inserta un objeto en la ventana de edición, y se nombra con "M4L.api.VideoThereminDetectAndDisplay" automáticamente se abre el subpatch con las dos salidas, en este caso "s …val1" va dirigida al knob pan, y la salida "s …val2" al knob freq. Antes de ser escalados en los rangos necesarios.

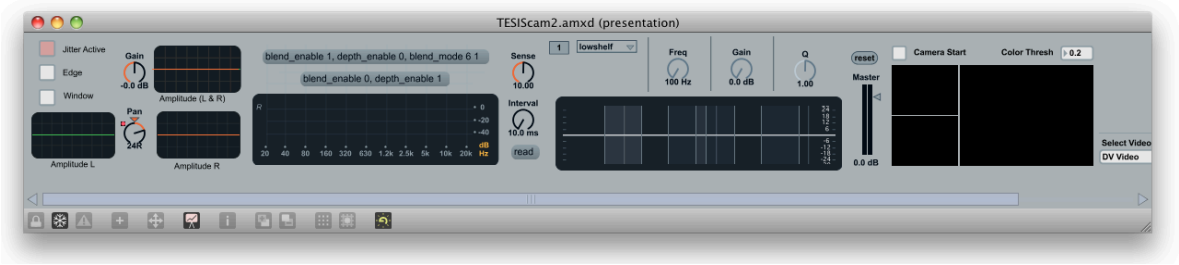
La programacion de el algoritmo "M4L.api.VideoThereminDetectAndDisplay" esta contenida en el ANEXO 2.

El empleo de este algoritmo únicamente podra trabajar en el canal habilitado con la cámara, no podrá ser usado en mas de un canal simultáneamente, para usarlo canal por canal se debe desactivar la cámara en los canales que no se desee esta herramienta, y activarla en el canal que se desee controlar.

5. PRESENTACION Y ANÁLISIS DE RESULTADOS

A través del proceso descrito anteriormente se llego a la creación de un sistema de procesamiento virtual VST cuyo propósito es lograr una transformación de la señal de sonora en un entorno grafico que facilita la espacialización de una mezcla de audio, mediante 5 procesos diferentes por canal. El primero es la caracterización del sonido por cada canal a través de un ecualizador de entrada, el segundo es el muestreo de esta señal por medio de un analizador de frecuencia *FFT (transformada rápida de fourier),* el tercero es un control y analizador del balance estereofónico (paneo), el cuarto es un control y analizador de amplitud, y el quinto es un transformador "audio-visual" de las variables descritas en los procesos anteriores. La interface grafica de usuario GUI del algoritmo para mezcla basado en diagramas de Gibson, diseñada por el autor del proyecto puede ser visualizada en la figura 42. Esta interfaz de usuario es creada en el mismo entorno de programación de *Max Msp* que encaja adecuadamente con el ambiente visual de *ableton live*.

Fig. 42. Plug-in para mezcla en diagramas de Gibson.



Las especificaciones del hardware y software donde se ha probado el funcionamiento del proyecto se explican a continuación. Para garantizar los resultados se han hecho las pruebas con bastante exigencia de hardware, es decir alto consumo de CPU y proceso gráfico.

Nota:

Las características para el proceso grafico se pueden simplificar, dejando la

ventana de visualización en la opción activa Edge, que muestra los vértices de las figuras, no cargando una imagen de fondo, y desactivando transparencias.

Fig. 43. Especificaciones de hardware para pruebas.

Sistema operativo Mac Os x versión 10.6, procesador de 1,8 GHz Intel Core 2 Duo, memoria DDR2 SDRAM 2 GB a 800 MHz.

El proceso de programación de este proyecto se inicio en el mismo computador pero con sistema operativo Windows 7 ultimate, con el fin de comprobar el funcionamiento en los dos sistemas operativos Windows y Mac, se continuo la programación en plataforma Mac Os x, importando los avances realizados y pruebas finales en este entorno sin ningún problema o inestabilidad.

la disposición de memoria RAM y de CPU es indispensable para el correcto funcionamiento del plugin en live, ya que incluye proceso de imagen estas exigencias son necesarias, las siguientes figuras muestran el consumo de

recursos del computador, cuando son activos los siguientes plugins en una sesión de live con 7 canales, como se muestra en la figura 44.

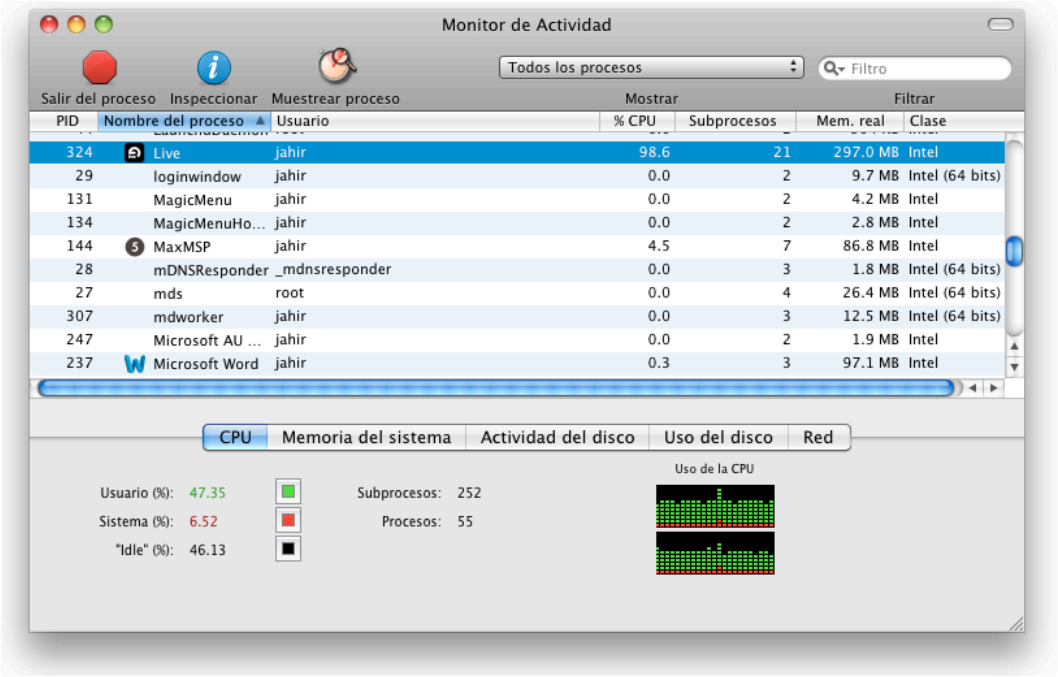
Fig. 44. Sesión de live con 7 canales activos.

En esta sesión de prueba en live, están activos los siguientes canales: Drums que es un bus que contiene al canal Kick, Snare y Hi hats, estos tres canales poseen cada uno como plugin, un sampler un compresor y el visualizador de Gibson, los siguientes canales son Bass, Synth y Pad, cada canal de estos contiene dos plugins adicionales al visualizador de Gibson, como compresores y filtros, es decir que en esta sesión que viene incluida en live por defecto, se le han agregado 5 plugins de visualización de Gibson por cada canal, y el consumo de sistema se muestra en los siguientes diagramas.

Fig. 45. Consumo de CPU desde el entorno de live.

En live en la parte superior derecha aparece una barra donde muestra el consumo total con todos los plugins activos y en reproducción como se ve en la figura 45.

Fig. 46. Monitor de Actividad con la sesión de live activa.



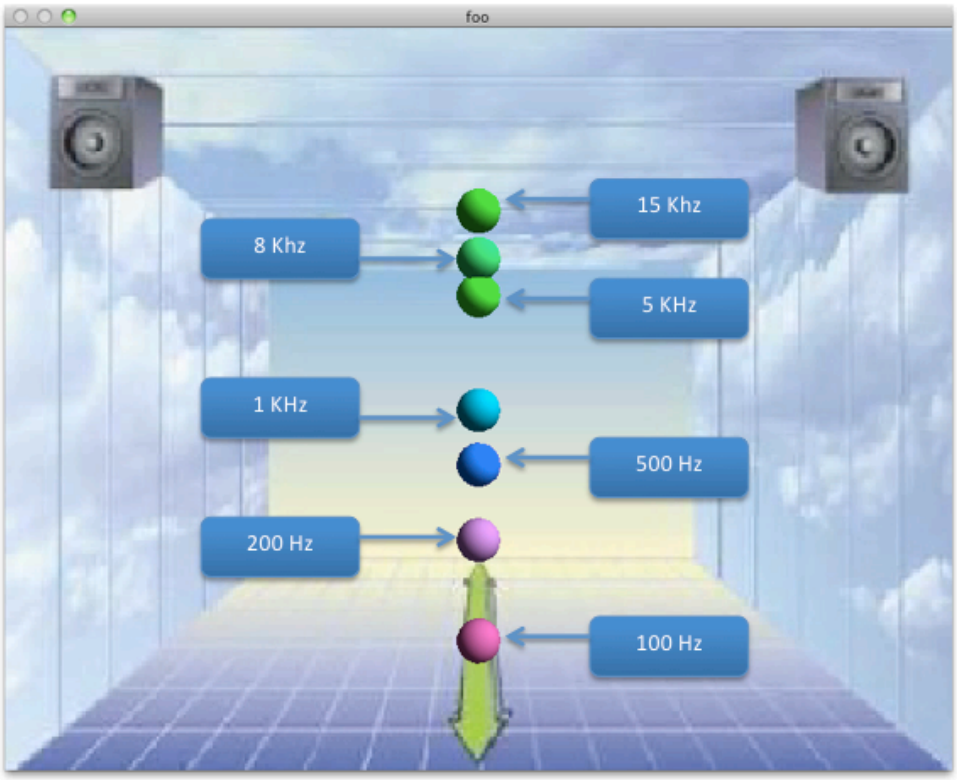
La aplicación de monitor de actividad de Mac OSX, muestra el consumo de recursos mientras esta activa la sesión de prueba anterior en funcionamiento.

El porcentaje de CPU (47.35%) es aproximado al analizado por live (43%), solo hay una diferencia de aproximadamente 4.35%, el uso de memoria ram por parte de *live* es de 297 MB y *Max Msp* 86.8 MB, es decir que los análisis del consumo de hardware en esta sesión de live con altas exigencias de proceso y varios plugins utilizados no alacanzan a superar el 50% de la capacidad de rendimiento en las pruebas de esta maquina. Por lo tanto se puede concluir que con estas especificaciones se pueden ubicar un numero aun muy superior de este plugin en distintos canales de una sesión de live.

5.1 Analisis De Color Mediante Tonos Puros.

Mediante la inserción de tonos puros por canal y la asignación del plugin del proyecto, se pueden comparar los resultados de la relación frecuencia color con la teoría expuesta en el libro de David Gibson. Las frecuencias utilizadas en esta prueba son: 50 Hz, 200 Hz, 500 Hz, 1000 Hz, 5000 Hz, 8000 Hz y 15000 Hz. La disposición vertical esta acorde a las frecuencias que se manejaron en esta prueba, por esta razón no están separados los objetos simétricamente.

Fig. 47. Análisis de color del plugin.

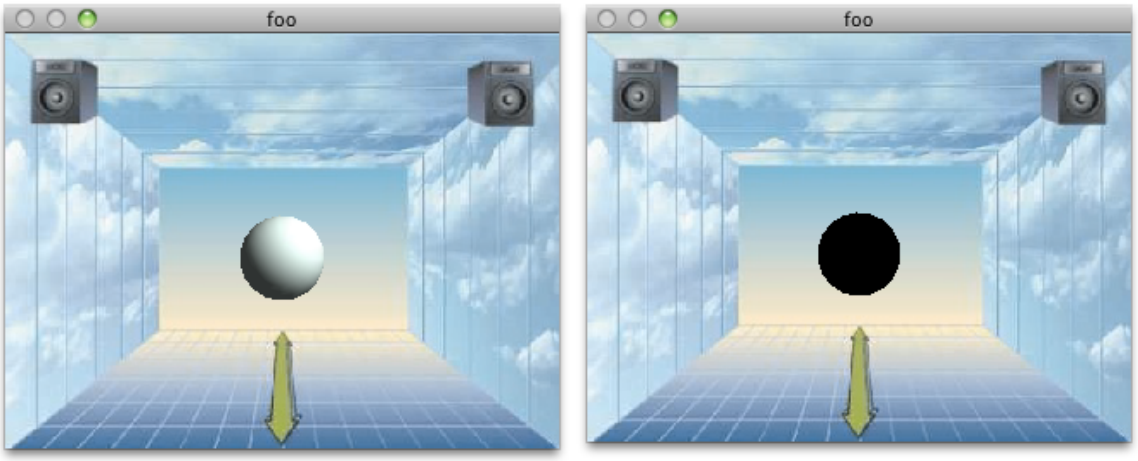


A continuación en la figura 48. Se pueden observar la teoría del color de David Gibson contra los resultados obtenidos desde la sesión de Live en la ventana de visualización del plugin. En la parte izquierda esta la teoría de David Gibson de la relación color-frecuencia, las frecuencias están entre rangos de octavas, y en la parte derecha esta la imagen de la ventana de visualización del plugin en funcionamiento, junto con la relación del tono puro al cual se a expuesto cada canal y la altura correspondiente a esa frecuencia, el color es determinado por el analizador de fft el cual asigna un color dependiendo la frecuencia.

Fig. 48. Comparación de relación Color-Frecuencia entre teoría y el plugin.

Finalmente se observa que los colores expuestos en la teoría concuerdan aproximadamente con los programados en el algoritmo, y por tanto si esta teoría es completamente valida se puede predecir que para un ruido blanco, el cual tiene igual contenido energético en todas las frecuencias, el color que se asignaría seria el blanco, y de acuerdo a lo expuesto en el diseño de la relación de color con frecuencia, cuando se suman todas las componentes RGB (Red Green Blue) el resultado es el color blanco. Así mismo cuando no hay sonido en el canal donde se ha asignado el plugin es decir no hay presencia de ninguna frecuencia el objeto "esfera" corresponderá con un color negro.

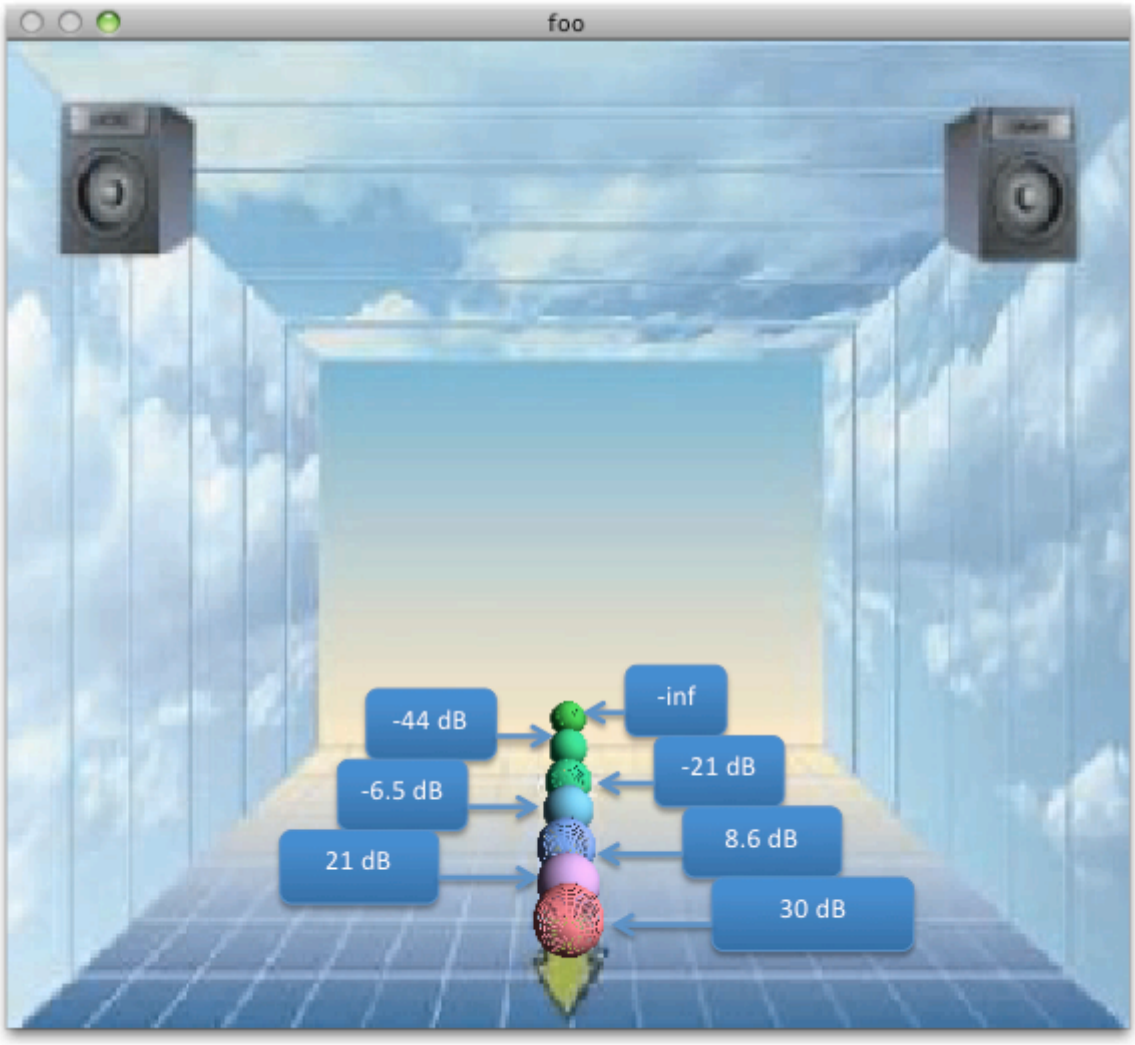
Fig. 49. Análisis de ruido blanco y ausencia de sonido.



5.2 Analisis De Volumen Y Paneo.

Mediante la misma sesión con los tonos puros se realiza una distribución de profundidad por medio del knob gain, para observar los niveles de volumen y su posición en la ventana de visualización, este análisis fue realizado en todo el rango programado para el volumen en el plugin.

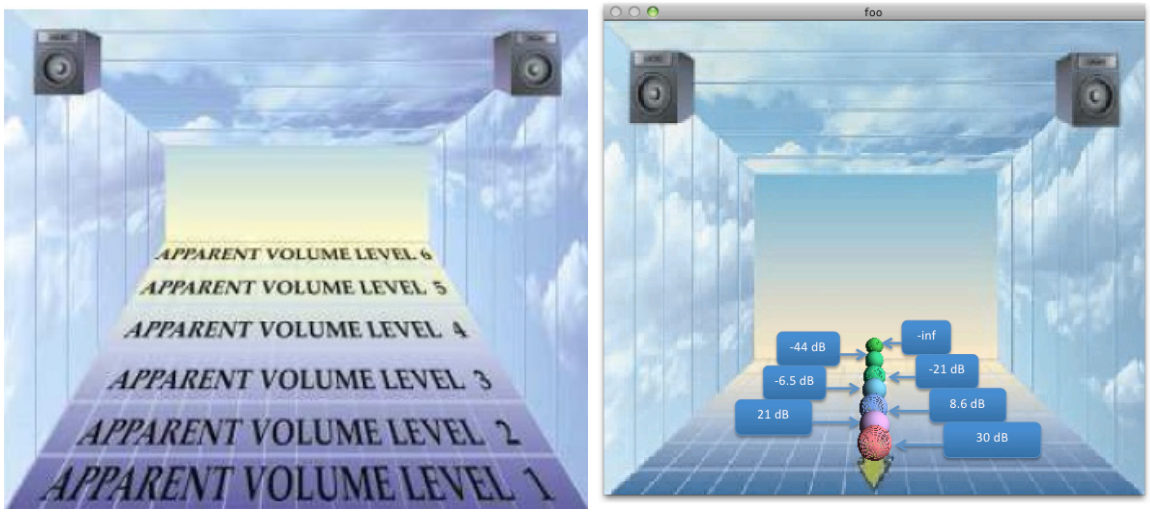
Fig. 50. Análisis de volumen del plugin.



Los niveles de volumen están distribuidos simétricamente, es notable la profundidad de los objetos por su radio, mientras menor sea su volumen, el radio de sus esferas será menor, algunos objetos mantienen activa la opción “edge” que

actúa como una transparencia, como ejemplo la esfera mas cercana, mantiene un renderizado diferente a la que le sigue para poder diferenciarlas. Estos valores en decibeles son obtenidos desde el knob gain de la izquierda en la figura 42. Este objeto viene como preset desde max for live, y sus valores de dB están en el rango entre (-inf y 30dB).

Fig. 51. Relación de niveles de volumen entre teoría y plugin.



De acuerdo con la teoría de David Gibson y los resultados obtenidos en la prueba del proyecto tenemos una relación entre teoría y el plugin. En la siguiente tabla se puede observar la relación entre los valores.

Tabla 2. Relación de valores de volumen con volumen aparente.

Teoria Volumen aparente	Valores finales del plugin en dB
Nivel 1 de volumen aparente	30 dB
Nivel 2 de volumen aparente	21 dB
Nivel 3 de volumen aparente	8,6 dB
Nivel 4 de volumen aparente	-6.5 dB
Nivel 5 de volumen aparente	-21 dB
Nivel 6 de volumen aparente	-44 dB

Los últimos valores que se observan a la derecha en la figura 51, son valores – inf,

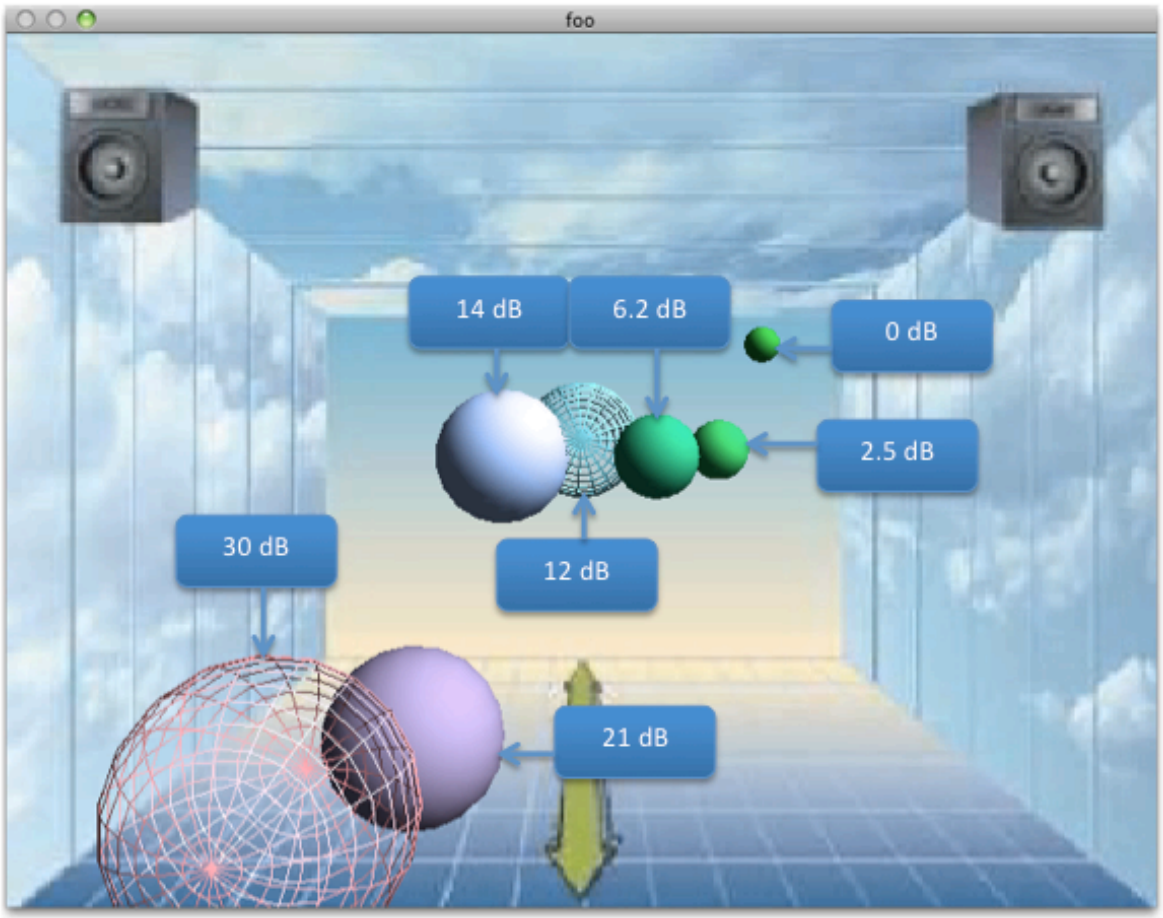
estos valores no tienen ninguna amplitud, pero sin embargo deben estar visualizados para saber la ubicación del objeto en cualquier momento.

Fig. 52. Análisis de paneo del plugin.

El rango de valores de paneo cubre todo el espacio estereofónico, en los extremos horizontales de la ventana de visualización se observan los valores de 50L para el limite izquierdo y 50R para el limite derecho, los valores frecuenciales son los mismos de las pruebas anteriores dispuestos de menor frecuencia a la izquierda y mayor frecuencia a la derecha.

Continuando con el análisis de resultados para el parámetro de paneo confirmamos que, los valores que controlan el radio de las esferas son dependientes del Knob Gain del ecualizador de entrada, debido a que este parámetro aumenta el nivel en la frecuencia que se a determinado, implícitamente este valor aumenta la amplitud espacial del objeto. A continuación en la figura 53 se muestran unos ejemplos de la variación de este parámetro con sus respectivos valores en decibeles.

Fig. 53. Análisis radial de las esferas en el plugin.



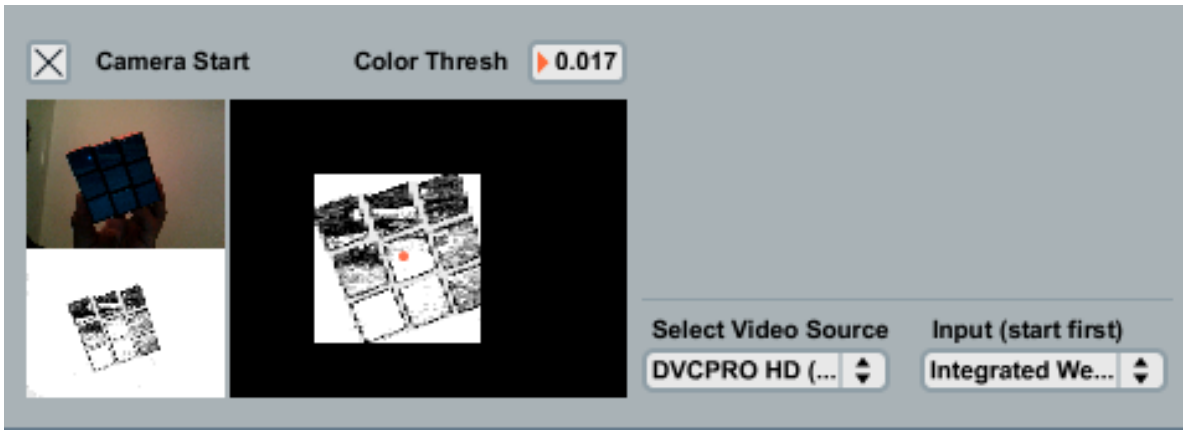
La disposición espacial de este análisis, fue aleatoria con el fin de mostrar las posibilidades de visualización de los objetos, cuando se encuentran ubicados compartiendo un mismo espacio, los radios de las esferas van dispuestos de

mayor a menor, es decir de mayor ganancia de la frecuencia en el ecualizador de entrada a menor ganancia, igualmente se utilizaron los tonos puros de la sesión de las anteriores pruebas, manteniendo las frecuencias bajas desde izquierda a derecha, como se vio anteriormente en el análisis del volumen, el radio parece variar según la profundidad visualmente, por la tridimensionalidad, pero esto es únicamente una apariencia por la perspectiva. El radio únicamente es controlado por la ganancia en el ecualizador de entrada.

5.3 Analisis Del Control Mediante Camara.

Finalmente se prueban los resultados de la parte final del algoritmo de este proyecto, el control de movimiento de los objetos vertical y horizontalmente por medio de una cámara, el proceso de detección de movimiento de este algoritmo reconoce un color determinado por el usuario.

Fig. 54. Control por movimiento.



En la ventana superior izquierda de la figura 54, se hace click en el color deseado para la captura de movimiento, al realizar esta acción en la venta inferior izquierda se filtran todos los colores diferentes al color seleccionado, y en la ventana derecha es observa la detección de movimiento por medio de un punto

central de color naranja, en este punto se enfocará la cámara para la detección y variación de parámetros que accionaran las partes del algoritmo general expuestas en el desarrollo ingenieril, en la parte superior izquierda, se observa un botón que activa y desactiva el manejo del plugin por cámara, adicionalmente hay un control numérico “Color Thresh” que permite variar el radio de captura de movimiento de la pantalla, es decir, mientras mayor sea el numero menor será la capacidad de reconocer el color deseado para el movimiento, y mientras menor sea el numero, mas estricto será el filtro del color, lo cual permitirá un mejor control del plugin, es necesario mantener el funcionamiento de esta sección del plugin, en un ambiente donde no halla demasiado contraste de color, es decir, con un solo fondo definido, y un color que se diferencie bastante del fondo.

6. CONCLUSIONES

El diseño de estos dispositivos virtuales “plugins”, no se limita únicamente a mejorar la calidad sonora o simular instrumentos tradicionales. Por esta razón fue necesaria la realización de este proyecto, como pauta de inicio en otra clase de plugins que no solamente mejoren la calidad sonora, sino que faciliten la composición general de la mezcla, aunque actualmente no existen muchos plugins que recreen la mezcla visualmente o no se puedan adquirir comercialmente, es muy probable que en un futuro no muy lejano, los cambios en la programación de estos dispositivos avancen paralelo a la evolución de la música informática. La piedra angular radica en el uso de un software que permita utilizar estos plugins en todas las estaciones de trabajo para difundir la información y la idea en la recreación virtual de la mezcla de audio.

El plugin desarrollado en este proyecto puede ser manipulado en ambos sistemas operativos, Windows y Mac OsX, pero solo en estaciones de trabajo de ableton live, característica de las ultimas actualizaciones de Max. Debido al cambio de *pluggo*[10], que permitía trabajar los plugins en cualquier estación de trabajo pero solo en plataforma Windows o Mac, a diferencia de Max for live.

Se logró recrear virtualmente la disposición espacial sonora desde la mezcla, aprovechando las ventajas de un software (Max Msp) capaz implementar todos sus patch en una estación de trabajo como ableton live, y automatizaciones en tiempo real. Estas arquitecturas sonoras virtuales describen y recrean los ambientes que se plantean desde la parte visual, generando así una correspondencia entre lo que se está viendo y lo que se oye.

Al realizar una correlación entre los colores obtenidos en la visualización del plugin y los expuestos en la teoría se observa una concordancia en la mayoría de las frecuencias, únicamente en el rango superior a 15 KHz David Gibson sugiere colores cercanos al amarillo, el plugin interpreta las frecuencias superiores a 15

[10] Pluggo: software incluido en versiones anteriores a MAX/MSP 5 que funciona como plataforma de exportación a VST de los *patchers* realizados en este programa.

KHz en la gama de color verde, ya que se maneja el formato (RGB). Asignando Rojo a Bajas, Azul a medias, Verde a altas frecuencias y mezclando estos colores dependiendo las amplitudes por cada frecuencia.

El rendimiento del plugin fue probado con bastantes exigencias en imagen y proceso de audio, en las sesiones expuestas en el análisis de resultados. En los prototipos implementados fué probado su complejidad/rendimiento con resultados óptimos que califican al plugin a operar con un bajo requerimiento de memoria física y virtual de la CPU.

El formato de creación GUI (Graphical User Interface) de Max for Live permite crear una interface de usuario intuitiva, llamativa, adecuada al entorno de live, y fácil de usar por múltiples usuarios ya que estos sistemas virtuales pueden ser distribuidos mediante medios y formatos digitales masivos como la Internet o discos compactos.

Se puede observar que al distribuir visualmente la mezcla de modo que no se sobrepongan las imágenes, auditivamente da una correspondencia con la posición espacial en la ventana tridimensional, por consecuencia se puede percibir una buena distribución en el espacio estereofónico para evitar el solapamiento de los instrumentos o canales.

En un futuro es posible, la realización de una mezcla sin ningún mando o interfaz como un teclado, un mouse u otro dispositivo controlador, únicamente con una interacción corporal, que permita mas sensibilidad entre la maquina y el hombre con el fin de realizar un mezcla de alguna manera mas manipulable. Además de la implementación en los actuales plugins que simulan instrumentos para que sea posible la interacción corporal como por ejemplo en este caso, es posible usar la cámara y el plugin para simular la interpretación de un theremin.

La teoría que expone David Gibson en su libro, contiene una amplia gama de conocimientos respecto a la mezcla, desde preproducción hasta postproducción, como imaginársela con respecto a numerosos efectos que existen actualmente, y

propone muchas soluciones para concluir en buenos resultados, iniciando desde la calidad en equipos e instrumentos, los conceptos básicos musicales que debe tener un ingeniero de sonido y hasta la forma de manejar relaciones personales con base a las sugerencias en arreglos de conceptos musicales con los artistas.

7. RECOMENDACIONES

Observando que progresa y los software para diseño de plug-ins poseen mejores herramientas, se recomienda ampliar los conocimientos en Max/MSP y no solo programas como C++ o Matlab, ya que es un software de programación con amplias aplicaciones en el diseño de creación de plug-ins de audio entre otros.

Se recomienda continuar con mejoras en el diseño del plugin como implementar el análisis e interpretación visual de otros efectos, como procesadores de tiempo, compresores, limitadores y muchos mas, ya que David Gibson sugiere una ayuda en su libro para interpretar visualmente estos parámetros.

Para solucionar el problema de amarillo en altas frecuencias, se sugiere ubicar las siguiente disposición de color-frecuencia: Rojo para frecuencias bajas, Azul para frecuencias medias bajas, Verde para frecuencias medias altas y Rojo para frecuencias altas, se sugiere de nuevo colocar rojo en frecuencias altas, ya que la sumatoria entre frecuencias medias altas y altas es decir entre rojo y verde, puede generar tonos amarillos.

Si se desea un manejo completo del plugin, desde la cámara, ya que cada dispositivo puede capturar dos variables, es posible copiar el algoritmo de la cámara y asignar un dispositivo externo para tener 4 variables, y así ser posible la asignación, de: Volumen, Paneo, Frecuencia, y Radio de los objetos “ganancia en el ecualizador de entrada”.

Mirar la posibilidad de importar la programación de Max for Live, en alguna versión anterior de Max, la cual por medio de *pluggo* permite exportar en un formato de VST que pueda funcionar en cualquier estación de trabajo.

Trabajar en un entorno de mezcla 5.1 y acomodar el plugin para lograr una espacializacion mas real en relación a la ventana de visualización y

aprovechar las herramientas visuales para logra una mejor mezcla.

Se recomienda trabajar el plugin mínimo con las especificaciones de hardware sugeridas en este documento para un correcto funcionamiento, ya que se ha probado en computadores con menores características y en algunos casos presenta bloqueo de las aplicaciones.

BIBLIOGRAFIA

Colasanto. Francisco, “Max/Msp: guía de programación para artistas”, 2009

Gibson. David, “The art of mixing”, 1997.

http://lib.tkk.fi/Diss/2001/isbn9512255324/article4.pdf

http://www.acis.org.co/index.php?id=319 “Software libre”

http://www.arts.ucsc.edu/pub/ems/maxtutors/

http://www.ccapitali.net/crc/tallermax/index.h

http://www.hispasonic.com/noticias/cycling-74-ableton-presentan-max-for-live/3554

http://www.maxmsp.es

http://www.puredata.info

http://www.uceva.edu.co/ingenieria/images/norma/ntc1486.pdf

Joyanes Aguilat L.; Fundamentos de programación: Algoritmos, estructuras de datos y objetos. McGraw Hill.2003

Ken G. Pohlmann.Principles of Audio Digital, cuarta edición. McGraw-Hill, 2000.

Max/Msp, Cycling74, http://www.cycling74.com/products/max5

PROAKIS G. John; Tratamiento digital de señales: Principios, algoritmos y aplicaciones, tercera edición

Sampieri Hernandez. Roberto C, “Metodologia de la Investigacion”,1997.

Udo Zolzer, Xavier Amatrain, “DAFX Digital Audio Effects”, 2002

V. Pulkki, “Generic panning tools for MAX/MSP”, http://www.lib.tkk.fi/Diss/2001/isbn9512255324/article4.pdf

ANEXO 1.

DAVID GIBSON MANUAL AL USUARIO.

En el dvd adjunto se encuentran el instalador de live 8 y Max versión demo para Mac Os X y Windows, una carpeta llamada David Gibson con12 plug-ins numerados, en estan también las imágenes que se pueden cargar como fondo de la ventana de visualización.

Despues de intalar los programas Live 8 y Max msp, se debe buscar la carpeta contenedora de los plugins para poder asignarlos en una sesión de audio.

En este ejemplo la carpeta de plugins se ha dejado en el escritorio para un facil acceso.

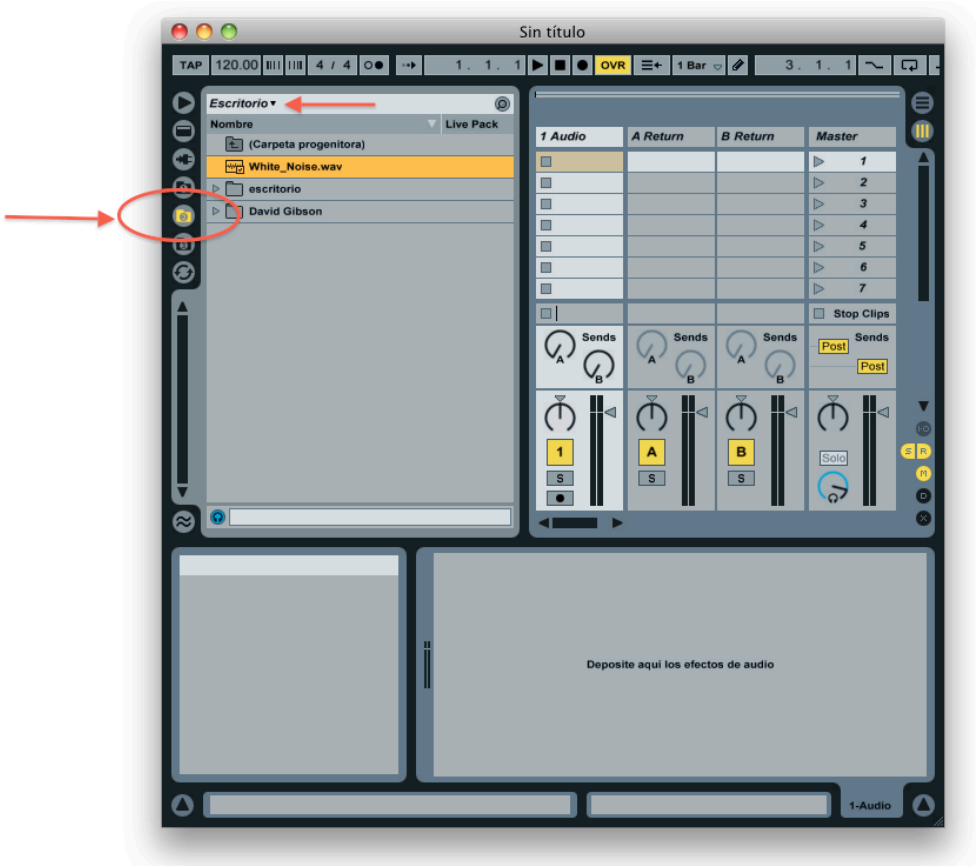


Fig A. Selección del directorio de los plugins.

a continuación despues de cargar una sesion de audio, se deben seleccionar por canal un unico plugin diferente y automaticamente aparece la ventana de visualizacion “foo” en fondo negro. Luego se debe cargar la imagen de fondo y activar jitter para que funcione el renderizado de los objetos.

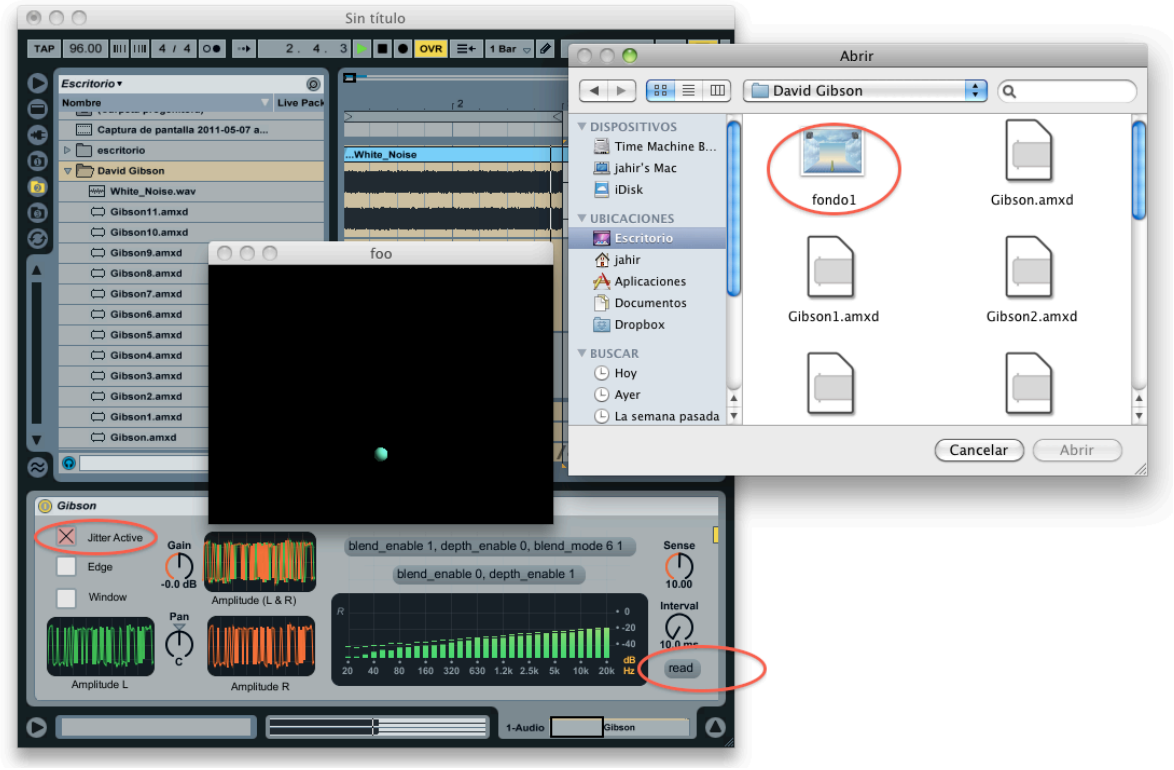


Fig. B. Activacion de la ventana de visualización

Para asignar los demas plugins, unicamente es necesario seleccionar los canales de audio en los cuales se desea visualizar y hacer doble clik, en cada plugin, despues de realizar la selección de imagen en un plugin y activar la opcion de jitter, no sera necesaria la repeticion de este proceso.

AUTOMATIZACIONES.

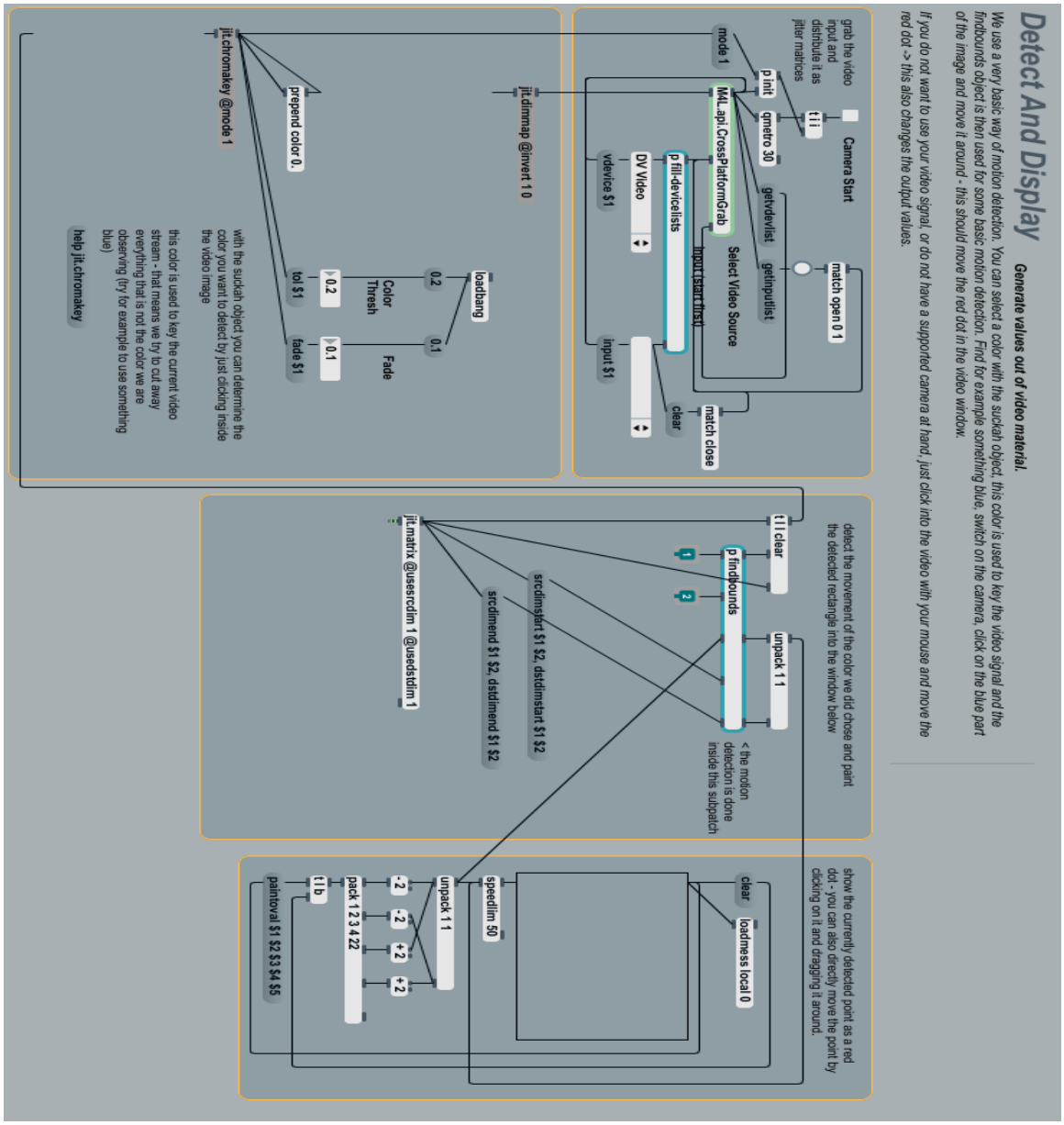
Para la asignación de las automatizaciones, en la ventana de edición, se deben buscar las opciones del plugin en la parte derecha del plugin, posteriormete en el menu desplegable de la parte inferior a la selección de automatizaciones, aparecen los parametros que se pueden modificar.

Fig. C. Selección de automatizaciones.

despues de asignar la automatización, la herramienta superior con forma de lapiz, debe ser activada y asi dibujar la automatización deseada dobre la onda sonora, automaticamente esta automatización influirá en la ventana de visualización haciendo posible un movimiento o cambio de los objetos.

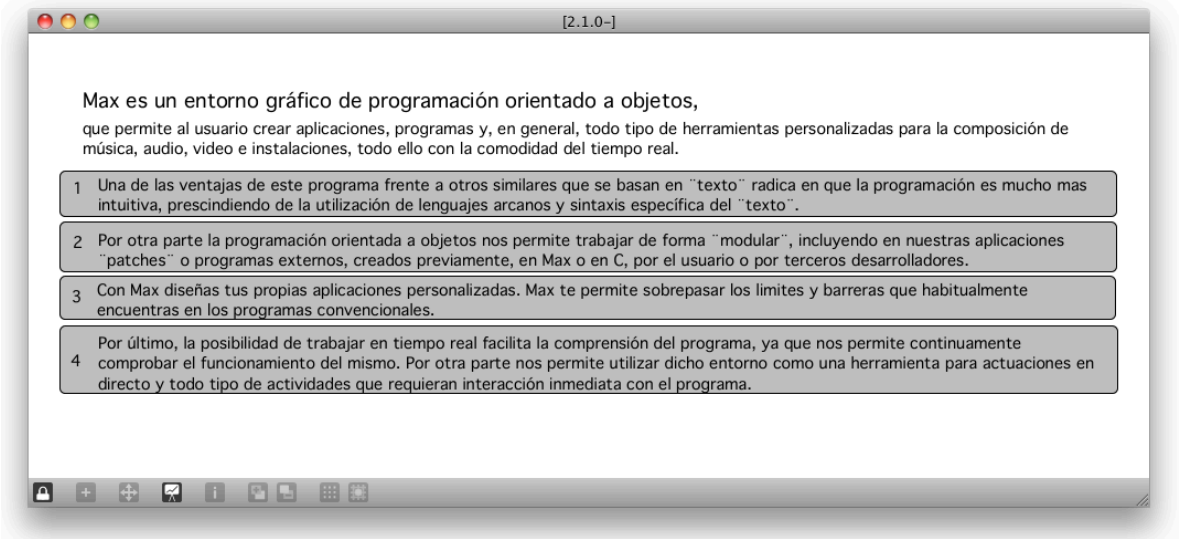
ANEXO 2.

PATCH “M4L.api.VideoThereminDetectAndDisplay”

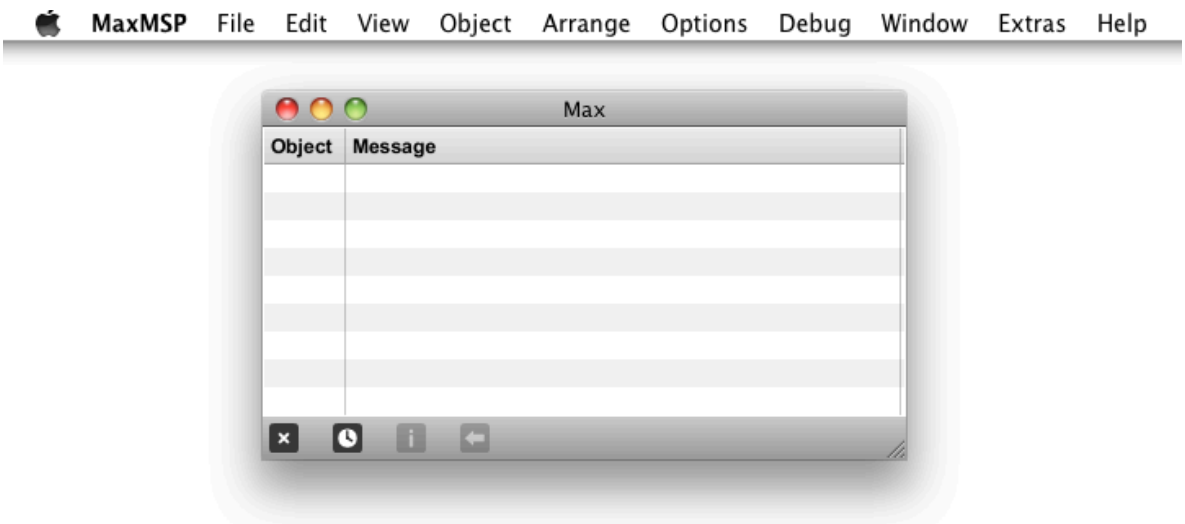


ANEXO 3.

TODO SOBRE MAX/MSP



VENTANA DE MAX



Lo primero que nos encontramos al abrir la aplicación es la ventana de Max, en la cual visualizaremos la información referente a:

Mensajes con información de algunos externals al iniciar el programa.

Mensajes referentes al driver de audio activo.

Mensajes de error debidos a conexiones no permitidas, externals no encontrados, etc.

El usuario no puede modificar los mensajes que aparecen en esta ventana, aunque si puede visualizar cualquier tipo de mensajes mediante el objeto -print-.

Para sacar la ventana de Max, ir a Menú - Windows - Max o pulsar el atajo cmd+m.

PATCHER

Es la ventana programación habitual. Para crear un patcher nuevo vamos a File - New - Patcher ó pulsamos cmd+n. El patcher está compuesto de:

- En la barra de titulo, de izquierda a derecha, las siguientes pestañas: cerrar ventana, edición patcher (cmd+click en espacio blanco o cmd+e), activar/desactivar MIDI en patcher, zoom y colapso de ventana, (Mac OS 9.x).

- La paleta de objetos (en la parte superior).

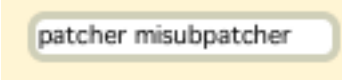
- Espacio en blanco para programar.

- La asistencia, (en la parte inferior izquierda, siempre que esté activada en Options - Asistence), nos muestra información de los nombres y funciones de las entradas y salidas de los objetos cuando nos situamos encima con el ratón.

Max es un programa de arquitectura modular, que nos permite encapsular patches y programas dentro de:

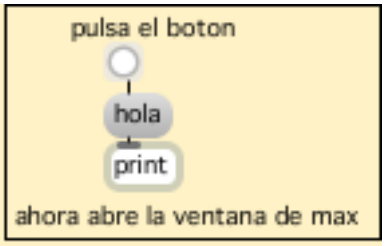
- Subpatchers;

Objetos cuyo nombre comienza por la palabra patcher seguido de un espacio en blanco y el nombre del subpatcher.



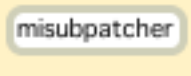
- Bpatchers;

objeto de interface de usuario que permite encapsular un patcher y tener acceso a los parámetros desde fuera del mismo.



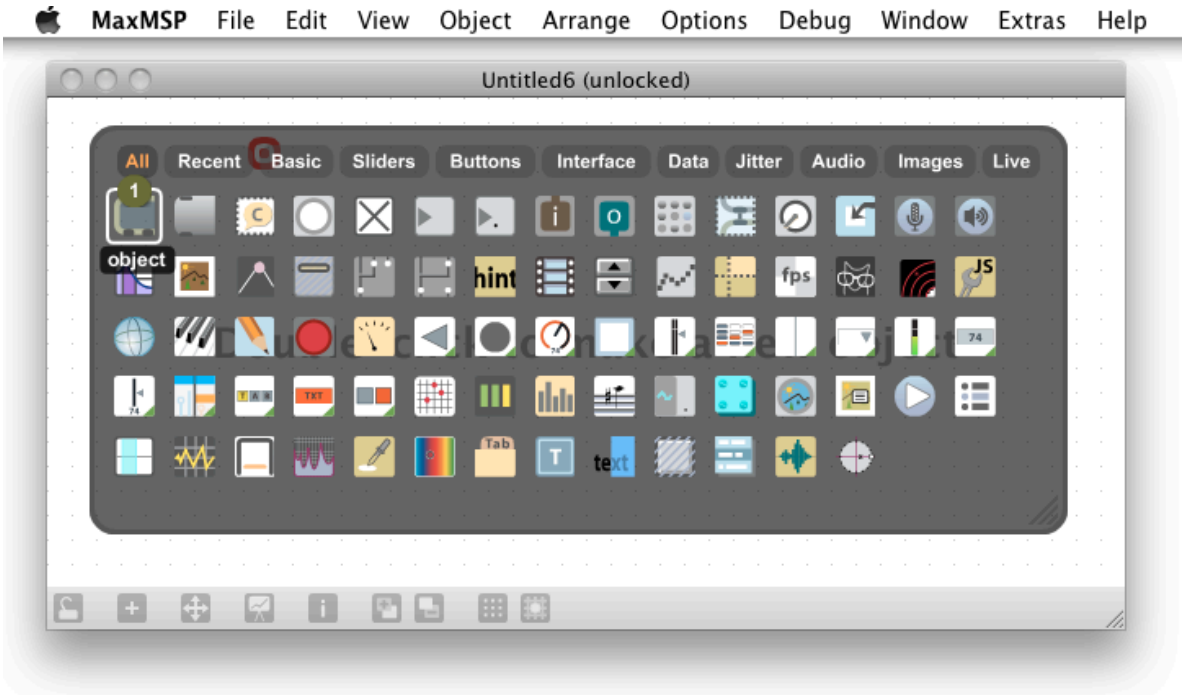
- Abstracciones;

son programas creados en el entorno grafico de Max, que pueden ser utilizados como cualquier otro external. Son llamados (siempre y cuando estén en el directorio adecuado) desde cualquier otro patcher.



Para desplegar el menú contextual del patcher pulsar en edición el botón derecho del ratón ó en su defecto ctrl+click en espacio en blanco

OBJETOS



Los objetos son la unidad básica de Max, se interconectan en el patcher mediante patch cords (conexiones), por donde circulan mensajes. Los objetos nos permiten realizan diferentes tareas mediante un sistema de entrada, proceso y entrega de la información. Todo ello se produce a intervalos de un milisegundo, (tiempo controlado por el reloj interno de Max - Max Scheduler), bien sea por nuestra intervención sobre el programa ó por la activación de una tarea del programador de eventos.

Podríamos establecer una primera distinción entre objetos: Built-in (que forman parte de la aplicación) y Externals (código C adaptado al entorno de Max), aunque en la práctica apenas se utiliza, ya que es imposible saber si un external ha sido creado como parte de la aplicación o no. Un ejemplo claro de ello lo tenemos en las librerías de MSP y Jitter, que contienen objetos externals, dedicados al audio y al video respectivamente, que fueron implementados para Max posteriormente.

TIPOS DE OBJETOS

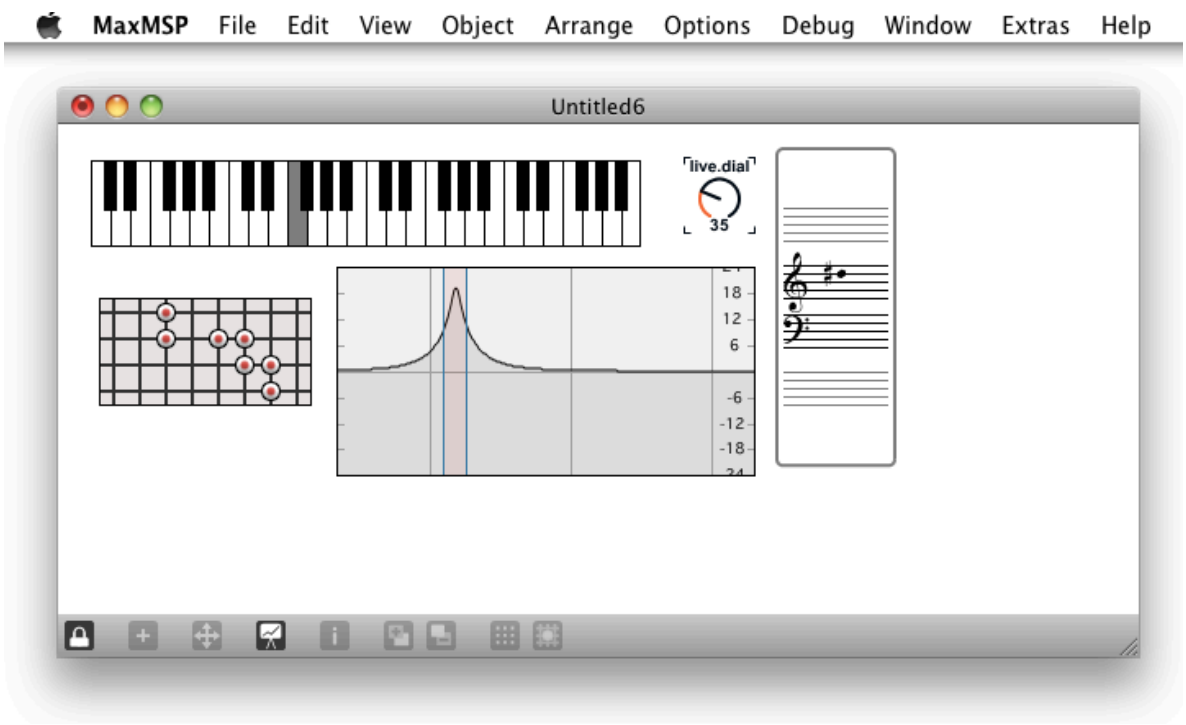
Normales, son aquellos que ¨cargamos¨ en el patcher en cajas rectangulares (Object Box). Arrastrando el primer icono de más a la izquierda en la paleta de objetos ó introduciendo directamente el nombre del external en una caja de objetos vacía. Para cargar un external en Max el programa tiene que conocer el

directorio donde se encuentra, (externals es la carpeta donde los busca por defecto), pudiendo asignar otro directorio de búsqueda en el Menú Options - File Preferences.

objeto cargado en el patcher correctamente.

objeto no cargado

UI, (User Interface), son los objetos gráficos que permiten al usuario accionarlos y modificarlos directamente con el ratón, MIDI, etc. Estos objetos se cargan al iniciar el programa desde al carpeta Max-Startup, y aparecen como iconos en la paleta de objetos en el patcher. Son importantes a la hora de construir interfaces personalizados.



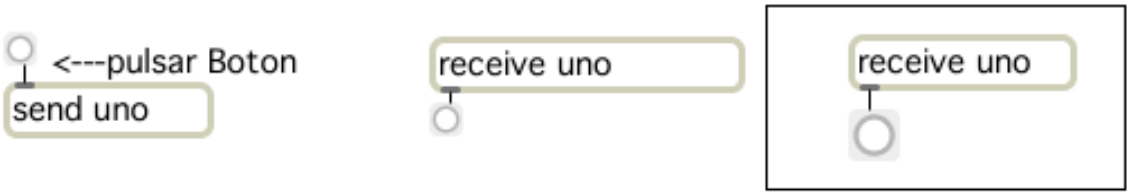
CONEXIONES

Las conexiones, tal como hemos apuntado en el apartado anterior, son las encargadas de comunicar los objetos entre sí, y por ellas circula información. Hay un tipo de conexión especial para los objetos MSP, que se diferencia, además de por la información que maneja (señales), por el color y tamaño (son amarillas y negras además de ser mas gruesas que las de Max). Más adelante hablaremos de ellas. Para realizar una conexión entre dos objetos en Max (siempre y cuando tengamos desactivada la opción Menú - Options - SEGMENTED PATCH CORD) seguiremos la regla de unir las salidas (outlets) con entradas (inlets). Una vez situamos el puntero encima de una salida, esta aumentará de tamaño y a su vez la asistencia nos mostrará que tipo de mensaje entrega el objeto, hacemos click y arrastramos hasta la entrada del siguiente objeto, que de igual manera aumentará su tamaño y mostrará en la asistencia la información del tipo de mensaje que admite.

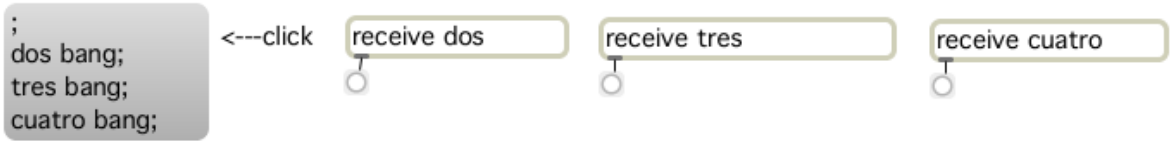
conexión entre dos objetos UI (boton e interruptor) conexión entre objetos MSP (phasor~ y cycle~)

Existen dos metodos basicos para hacer las conexiones entre objetos

Mediante los objetos send y receive. El objeto send seguido de un nombre lo vincula a un objeto receive con el mismo nombre, sin necesidad de realizar una conexión ¨física¨ entre ambos. Este sistema funciona entre subpatches, bpatcher y abstracciones.



Mediante una caja de mensajes (Messsage Box) de la siguiente forma: en la primera línea punto y coma, y en las siguientes líneas el nombre del objeto receive al que se quiere enviar el mensaje seguido del mensaje correspondiente.

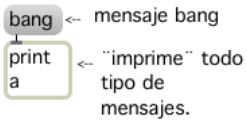


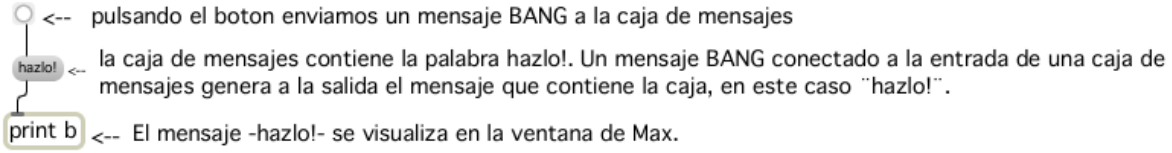
MENSAJES

Toda la información que circula entre las conexiones de los objetos en Max es un Mensaje.

Tipos de mensajes:

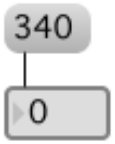
- BANG - Es un mensaje especial de Max que significa ¨Hazlo!¨





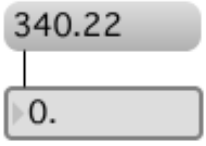
- MENSAJE NUMÉRICO: al hacer click sobre la caja de mensaje se genera a la salida un mensaje numérico entero.

- INT (Número Entero)



<-- el objeto UI (NUMBER BOX) está preparado para recibir/visualizar números enteros.

- FLOAT (Número Decimal) al hacer click sobre la caja de mensaje se genera a la salida un mensaje numérico decimal.



<-- el objeto UI (FLOAT NUMBER BOX) está preparado para recibir/visualizar números decimales.

- LISTA - Es un mensaje que contiene un conjunto de números (tanto enteros como decimales) separados por un espacio en blanco.

1 2 3 4.1 11 12 16 67 34 44100 512 65365

- SYMBOL - Es una palabra.

plug

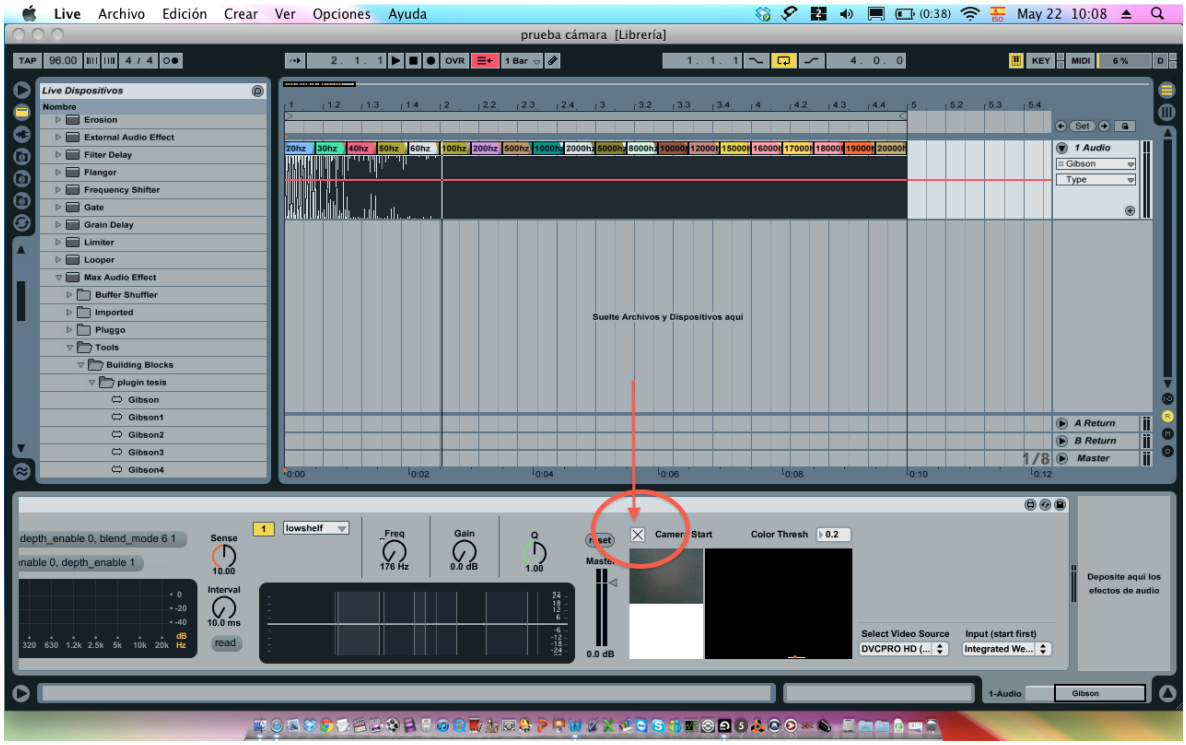
open

close

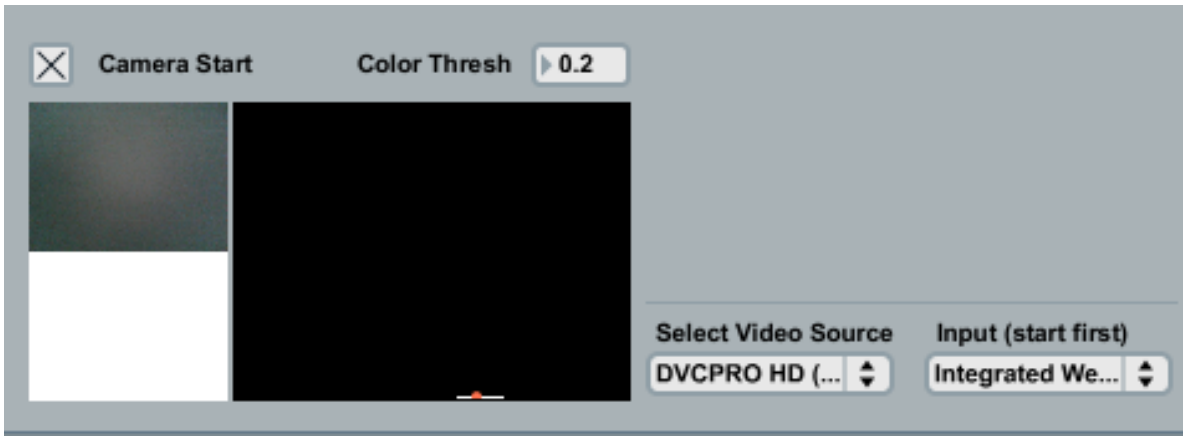
ANEXO 4.

EJEMPLO DE USO DE LA CÁMARA EN EL PLUGIN.

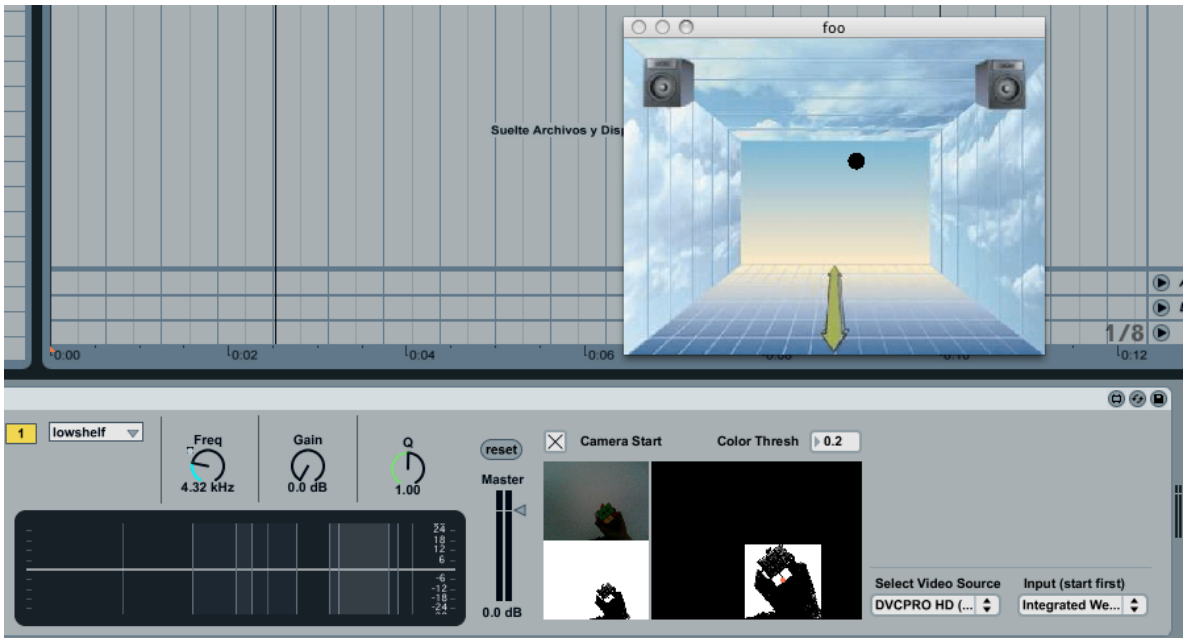
En una sesión ejemplo en ableton live, con un canal, en el cual se encuentra un barrido de frecuencia, se va a cargar el plugin de este proyecto, luego se habilitara la opcion de la cámara como se muestra en la siguiente figura.



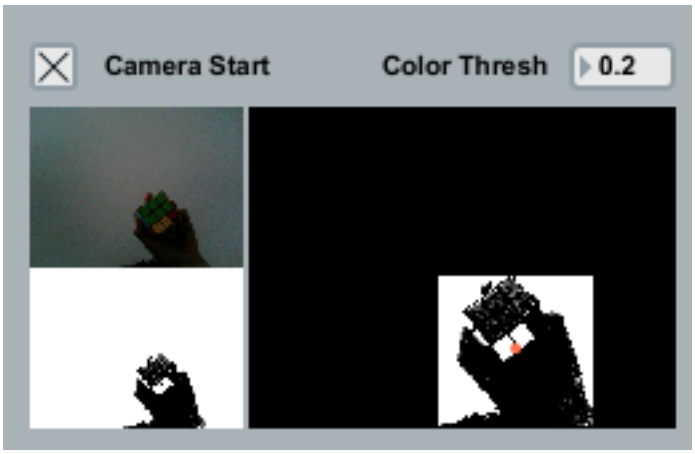
Después de activar la cámara, se visualizara lo que se este registrando en la ventana superior izquierda de la siguiente figura. Al realizar esta operación la cámara ya estar activa para live, y no se podra usar en ninguna otra aplicación mientras ya se encuentre mostrando lo que debe visualizar. Igualmente si no se ha activado, es necesario cerrar todas las posibles aplicaciones que la puedan estar utilizando, para unicamente dejarla en funcionamiento en live, además, la cámara solamente debe ser activada en cada canal donde se este trabajando el plugin, ya que no podra registrar simultaneamente mas de un canal.



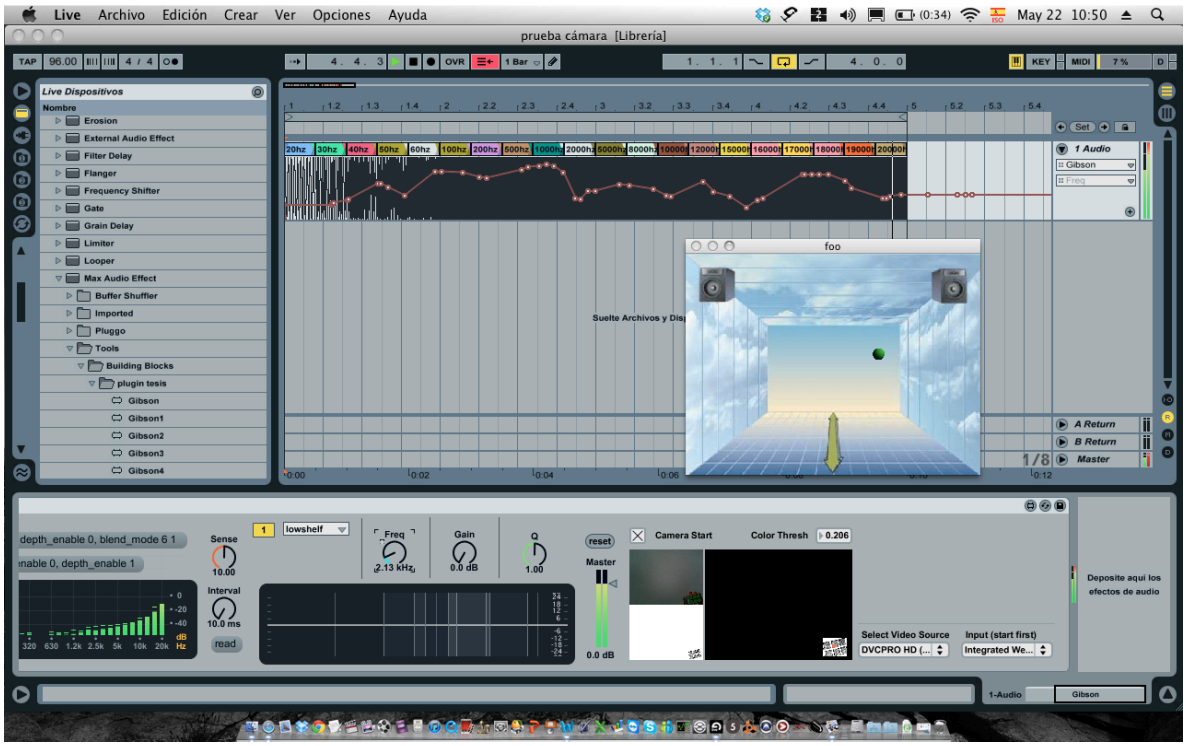
Luego de observar que la cámara esta registrando imagen, se procede a activar la ventana de visualización jitter, para visualizar el objeto “esfera” y además cargar la imagen de fondo de la ventana, ya que si no se realiza, el fondo sera totalmente negro.

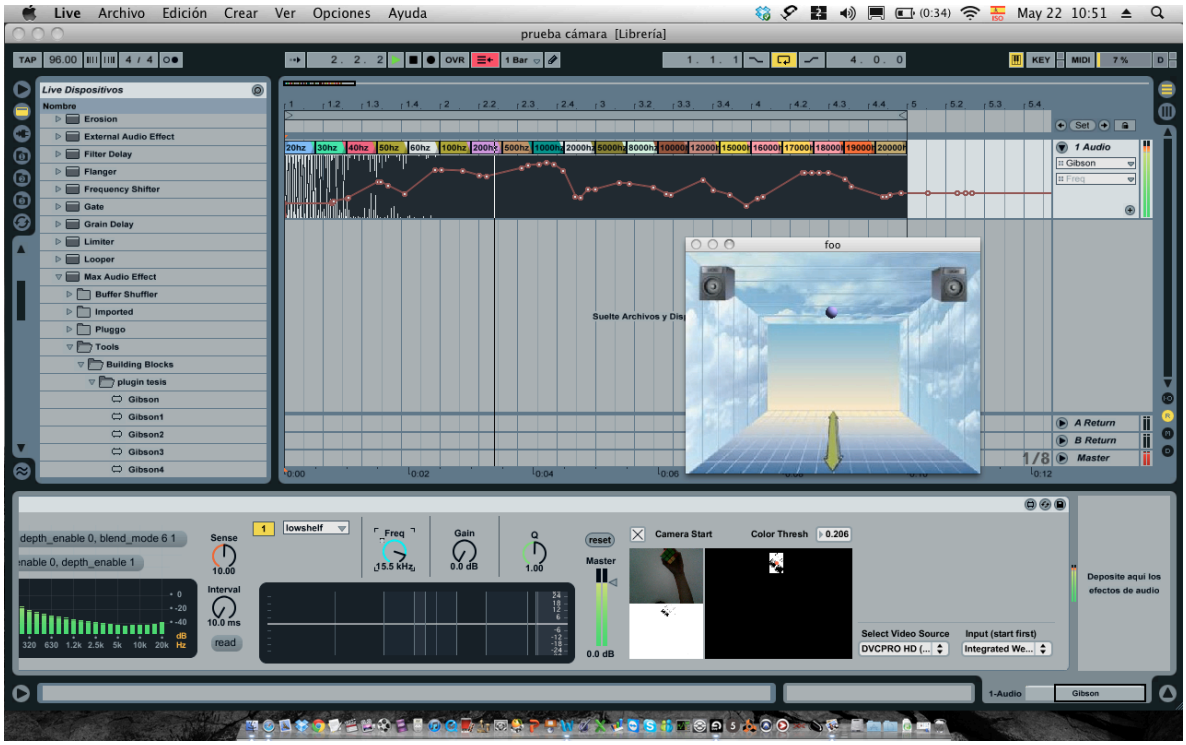


posteriormente se debe usar un objeto de algún color diferente al fondo en el cual se va a trabajar la cámara, para que esta misma puede diferenciar bien el objeto del resto de la imagen de fondo, en este ejemplo se uso un objeto de color verde.



al mover el objeto de color verde, automaticamente se varian los knobs, de paneo y de frecuencia en el ecualizador de entrada, por consecuencia la esfera en la ventana de visualizacion tambien se moverá dependiendo la direccion del objeto que la camara este registrando.





como se puede visualizar en las anteriores figuras, el objeto esfera en la ventan de visualización tiene correspondencia con el objeto verde que se ubico frente a la camar, e igualmente el knob frecuencia tambien varia respecto a este movimiento, adicionalmente se puede hacer la grabación de esta automatización, y el objeto esfera en la ventana de visualizacion despues de la grabacion, que dara dispuesto a la automatización del movimiento realizado por medio de la cámara.