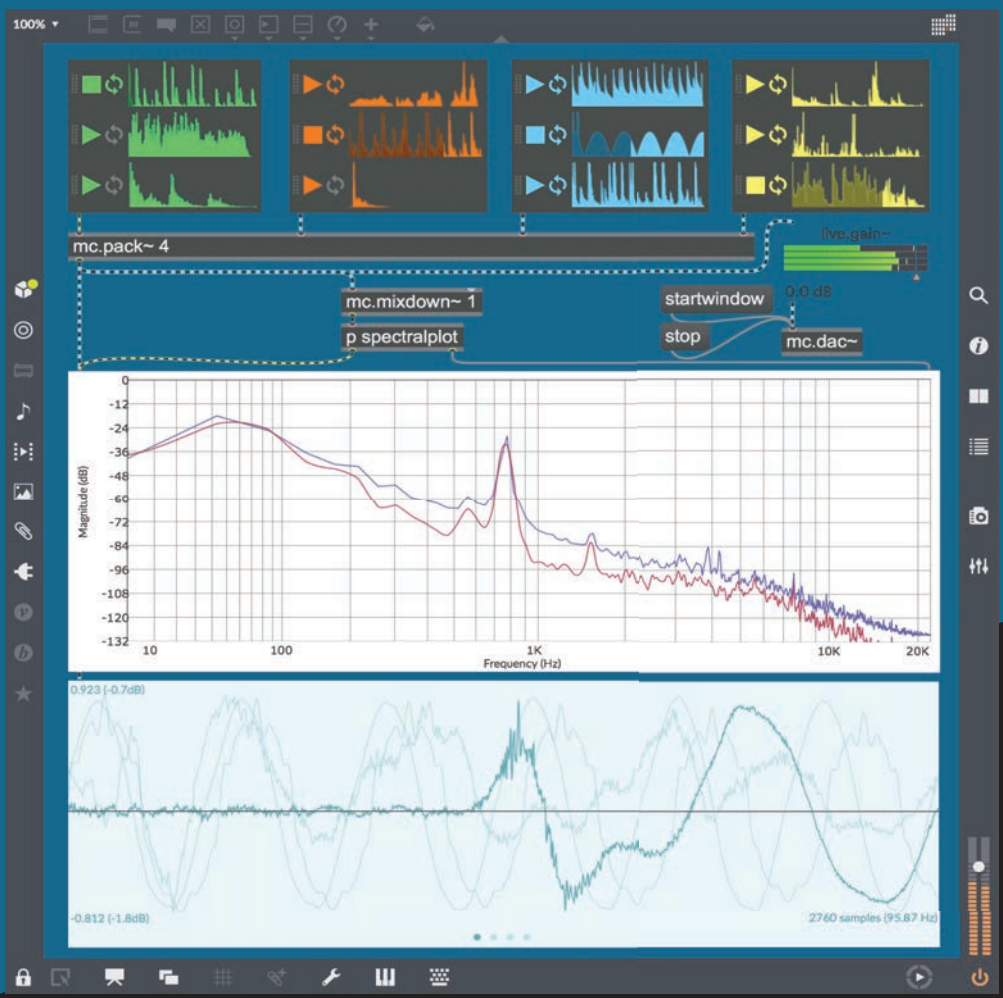
Alessandro Cipriani • Maurizio Giri



Música Electrónica y Diseño Sonoro

Teoría y Práctica con Max 8 • volumen 1

ConTempoNet

Alessandro Cipriani • Maurizio Giri

Este es un extracto del libro:

MÚSICA ELECTRÓNICA Y DISEÑO SONORO

Teoría y Práctica con Max 8 - Volumen 1

para más información:

www.contemponet.com
www.virtual-sound.com

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

CIPRIANI A. - GIRI M.
MÚSICA ELECTRÓNICA Y DISEÑO SONORO
Teoría y Práctica con Max 8
Vol. 1
ISBN 978-88-99212-17-9

Titulo original: Musica Elettronica e Sound Design - Teoria e Pratica con Max 8
Copyright © 2009-2019 Contemponet s.a.s. Rome - Italy

Traducción:
Maribel Orozco: Coordinación general y revisión de la traducción.
Aaron Montoya Moraga: Traducción de los capítulos de práctica y del Interludio A.
Maribel Orozco: Traducción de los capítulos de teoría y del Interludio B.

Copyright © 2020 - Contemponet s.a.s., Rome - Italy
Primera edición 2020

Diseño de imágenes: Gabriele Cappellani y Maurizio Refice
Realización de ejemplos interactivos: Francesco Rosati
Realización del índice: Salvatore Mudanò
Asesoría didáctico-lingüística: Damiano De Paola

Los productos y nombres de compañías aquí mencionados pueden ser marcas comerciales de sus respectivas compañías. La mención de productos de terceros es solo para fines informativos y no constituye una aprobación ni una recomendación. El uso de nombres descriptivos generales, nombres comerciales, marcas comerciales, etc., en esta publicación, incluso si los primeros no están especialmente identificados, no debe tomarse como una señal de que dichos nombres, según lo entiende la Ley de Marcas Comerciales y Marcas de Mercadería, puedan ser utilizados libremente por cualquier persona.

Reservados todos los derechos. Ninguna parte de esta publicación puede ser reproducida, almacenada, transmitida o utilizada en manera alguna por ningún medio, ya sea electrónico o mecánico, sin el previo consentimiento por escrito del editor.

Contemponet s.a.s. - Roma
e-mail posta@contemponet.com
posta@virtual-sound.com
URL: www.contemponet.com
www.virtual-sound.com

ÍNDICE

Notas sobre la traducción • VII
Prefacios • IX
Introducción • XIII

Capítulo 1T - TEORÍA
INTRODUCCIÓN A LA SÍNTESIS DE SONIDO

AGENDA DE APRENDIZAJE
1.1 Síntesis de sonido y procesamiento de señales
1.2 Frecuencia, amplitud y forma de onda
1.3 Variaciones de frecuencia y amplitud en el tiempo: envolventes y glisandos
1.4 Relación entre frecuencia e intervalo musical
1.5 Notas sobre el trabajo con sonidos sintetizados
1.6 Notas sobre panning
Conceptos fundamentales
Glosario

Capítulo 1P - PRÁCTICA
SÍNTESIS DE SONIDO CON MAX

AGENDA DE APRENDIZAJE
1.1 Primeros pasos con Max
1.2 Frecuencia, amplitud y forma de onda
1.3 Variaciones de frecuencia y amplitud en el tiempo: envolventes y glisandos
1.4 Relación entre frecuencia e intervalo musical
1.5 Notas sobre el trabajo con sonidos sintetizados
1.6 Notas sobre panning
1.7 Algunos fundamentos de Max
Lista de comandos principales de Max
Lista de objetos de Max
Lista de mensajes, atributos y parámetros de objetos de Max específicos
Glosario

Interludio A - PRÁCTICA
PROGRAMANDO CON MAX

AGENDA DE APRENDIZAJE
IA.1 Max y los números: los operadores binarios
IA.2 Generación de números aleatorios
IA.3 Manejo del tiempo: el objeto metro
IA.4 Subpatches y abstracciones
IA.5 Otros generadores de números aleatorios
IA.6 Orden de mensajes con trigger
IA.7 Objetos para administrar listas
IA.8 La message box y los argumentos variables
IA.9 Envío de secuencias de bangs: el objeto uzi
IA.10 Send y receive

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

IA.11 Uso de atributos dentro del objeto
IA.12 Audio multicanal
Lista de objetos de Max
Mensajes, atributos y parámetros para objetos específicos de Max
Glosario

Capítulo 2T - TEORÍA
SÍNTESIS ADITIVA Y SÍNTESIS VECTORIAL
AGENDA DE APRENDIZAJE
2.1 Síntesis aditiva de espectro fijo
2.2 Batimientos
2.3 Fundidos cruzados de tablas de ondas: síntesis vectorial
2.4 Síntesis aditiva de espectro variable
Conceptos fundamentales
Glosario
Discografía

Capítulo 2P - PRÁCTICA
SÍNTESIS ADITIVA Y SÍNTESIS VECTORIAL
AGENDA DE APRENDIZAJE
2.1 Síntesis aditiva de espectro fijo
2.2 Batimientos
2.3 Fundidos cruzados de tablas de ondas: síntesis vectorial
2.4 Síntesis aditiva de espectro variable
Lista de objetos de Max
Mensajes, atributos y parámetros de objetos de Max específicos
Glosario

Capítulo 3T - TEORÍA
GENERADORES DE RUIDO, FILTROS Y SÍNTESIS SUSTRACTIVA
AGENDA DE APRENDIZAJE
3.1 Fuentes sonoras para la síntesis sustractiva
3.2 Filtros pasa bajos, pasa altos, pasa banda y rechaza banda
3.3 El factor Q o factor de resonancia
3.4 Órdenes de filtros y conexión en serie
3.5 La síntesis sustractiva
3.6 Ecuaciones de filtros digitales
3.7 Filtros conectados en paralelo, y ecualización gráfica
3.8 Otras aplicaciones de conexión en serie: ecualizadores paramétricos y filtros shelving
3.9 Otras fuentes para síntesis sustractiva: impulsos y cuerpos resonantes
Conceptos fundamentales
Glosario
Discografía

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

Capítulo 3P - PRÁCTICA
GENERADORES DE RUIDO, FILTROS Y SÍNTESIS SUSTRACTIVA
AGENDA DE APRENDIZAJE
3.1 Fuentes sonoras para la síntesis sustractiva
3.2 Filtros pasa bajos, pasa altos, pasa banda y rechaza banda
3.3 El factor Q o factor de resonancia
3.4 Órdenes de filtros y conexión en serie
3.5 La síntesis sustractiva
3.6 Ecuaciones de filtros digitales
3.7 Filtros conectados en paralelo, y ecualización gráfica
3.8 Otras aplicaciones de conexión en serie: ecualizadores paramétricos y filtros shelving
3.9 Otras fuentes para síntesis sustractiva: impulsos y cuerpos resonantes
Lista de objetos de Max
Atributos para objetos de Max específicos
Glosario

INTERLUDIO B - PRÁCTICA
OTROS ELEMENTOS DE PROGRAMACIÓN CON MAX
AGENDA DE APRENDIZAJE
IB.1 Notas sobre MIDI
IB.2 El operador módulo y la recursión
IB.3 Rutear señales y mensajes
IB.4 Los operadores relacionales y el objeto select
IB.5 Descomponer una lista: el objeto iter
IB.6 Bucle de datos
IB.7 Generar una lista aleatoria
IB.8 Cálculos y conversiones con Max
IB.9 Uso de tablas para envolventes: el tono Shepard
Lista de objetos de Max
Mensajes y atributos para objetos Max específicos
Glosario

Capítulo 4T - TEORÍA
SEÑALES DE CONTROL
AGENDA DE APRENDIZAJE
4.1 Señales de control: el paneo estéreo
4.2 DC Offset
4.3 Señales de control para frecuencia
4.4 Señales de control para amplitud
4.5 Modulación del ciclo de trabajo (modulación por ancho de pulsos)
4.6 Señales de control para filtros
4.7 Otros generadores de señales de control
4.8 Señales de control: el paneo multicanal
Conceptos fundamentales
Glosario

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

Capítulo 4P - PRÁCTICA
SEÑALES DE CONTROL
AGENDA DE APRENDIZAJE
4.1 Señales de control: el paneo estéreo
4.2 DC Offset
4.3 Señales de control para frecuencia
4.4 Señales de control para amplitud
4.5 Modulación del ciclo de trabajo (modulación por ancho de pulsos)
4.6 Señales de control para filtros
4.7 Otros generadores de señales de control
4.8 Señales de control: el paneo multicanal
Lista de objetos de Max
Atributos para objetos específicos de Max
Glosario

Bibliografía

Índice

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

NOTAS SOBRE LA TRADUCCIÓN

El presente volumen del libro *Música Electrónica y Diseño Sonoro* ya ha sido traducido a lengua inglesa, y con la presente versión, los autores Alessandro Cipriani y Maurizio Giri se proponen llegar de manera más inmediata y eficaz a un vasto número de lectores de lengua española de todo el mundo, ya sean estudiantes de cursos relacionados con las nuevas tecnologías musicales, docentes de conservatorio y/o universidades, o simplemente aficionados, con algunos conocimientos sobre Max y que quieran profundizar en ellos, o incluso empezar desde cero.

Considerando que el español es una de las lenguas más habladas del mundo[1], y que es lengua oficial en 21 países, es natural que existan muchas variantes de lengua, caracterizadas, entre otros aspectos, por un uso particular de expresiones lingüísticas circunscritas en muchos casos, a algunas zonas geográficas de habla hispana.

En esta edición, los traductores han adoptado un modelo de lengua preferiblemente neutro, evitando caer en el uso de un léxico especializado típico de algunos países hispanohablantes, a favor del uso de vocablos de uso más difundido, no tanto a nivel de número de nativos, como de número de países . Es el caso de expresiones como *computadora*, cuyo uso se extiende a la mayor parte de países hispanoamericanos con excepción de Chile y Colombia, donde el vocablo predominante respectivo es *computador*; mientras que en España se prefiere el término *ordenador*.

Está también el caso de los anglicismos, de uso tan frecuente en textos técnicos como éste y que hacen referencia a las nuevas tecnologías. Considerando la proximidad de las naciones americanas hispanohablantes a los Estados Unidos, no solo en el aspecto geográfico sino también a causa la notable influencia que este país ejerce en Hispanoamérica a través de su política y de sus medios de comunicación, es de uso común oír expresiones que hacen alusión a términos técnicos en inglés, o incluso en un inglés que ha sido "españolizado", por así decirlo. Es frecuente entonces oír hablar de frecuencia de *sampleo* (por *sample*) en lugar de frecuencia de *muestreo* (por *muestras*); así como común es el uso de expresiones como *crossfade* (en lugar de *fundidos* cruzados), o *panning* (en lugar de *paneo* o espacialización). En éstos y en algunos otros casos, hemos optado por ser más fieles al uso de términos propios de la lengua española, entre otras cosas porque de por sí, el libro abunda en el uso de anglicismos del lenguaje técnico, típico de la materia de estudio de este libro, además de todo el léxico relacionado con el ambiente Max, para el cual se utiliza la lengua inglesa casi en su totalidad.

Por lo que se refiere a los vocablos técnicos que gozan de más de una versión (como *decibel/decibelio*; *sinusoidal/senoidal*; *auriculares/audífonos/cascos*), a menudo hemos optado por utilizar aquellos que presentaban una mayor frecuencia de aparición en páginas de internet, o en el banco de recursos CREA, de la página web de la RAE (*Real Academia Española*).

[1] Según datos del Instituto Cervantes, en 2017 el número de personas que tienen el español como lengua oficial llega a más de 477 millones, mientras que el grupo de usuarios potenciales del español supera los 572 millones ("El español: una lengua viva. Informe 2017" - Instituto Cervantes).

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

Otro aspecto importante que queremos señalar, es el sistema numérico usado para los separadores de millares y de decimales. Para los decimales, hemos usado el punto y no la coma, y para los millares, hemos creído conveniente no usar ningún tipo de separador, puesto que el Sistema Internacional de Unidades aconseja el uso de un espacio entre los dígitos, agrupando las cifras de tres en tres, pero para efectos de mayor claridad en este libro, no nos ha parecido adecuado. Por lo cual, para referirnos a la frecuencia de muestreo, por ejemplo, hemos escrito el número de esta forma: 44100, al igual que para las cifras superiores a 999, utilizadas en diversos ejemplos[2].
Ya para terminar, queremos hacer énfasis en que hemos tratado de minimizar al máximo otras diferencias de tipo morfológico que caracterizan las diferentes variantes de la lengua española, adaptándonos a un estilo narrativo neutro, de manera que cada lector, independientemente del lugar de procedencia, se sienta cómodo durante la lectura de este manual.

Los traductores,

Maribel Orozco - Aaron Montoya.

[2] Nótese que en algunos ejemplos de Max, cuyo ambiente se adapta al sistema de numeración anglosajón, es posible ver el uso de la coma como separador de millares.

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

PREFACIO
por David Zicarelli

Podrá parecerte extraño, pero hace muchos años, cuando leía libros y artículos para tratar de aprender a crear sonidos con la computadora, tenía que limitarme a imaginar cómo sonarían las técnicas de síntesis descritas. Si bien mi imaginación debe de haber sido estimulada por esta práctica, estoy feliz de que la tecnología haya avanzado hasta el punto de que hoy en día, prácticamente cualquier técnica de síntesis puede ser realizada en tiempo real con una computadora ordinaria. La experiencia perceptual, de hecho, es un elemento importantísimo en el aprendizaje de las técnicas de síntesis digital.

El libro de Alessandro Cipriani y Maurizio Giri es uno de los primeros cursos sobre música electrónica que integra explícitamente percepción, teoría y práctica, usando ejemplos de síntesis de sonido en tiempo real que pueden ser manipulados y personalizados. A mi parecer, la manipulación de sonido constituye un aspecto extremadamente importante en el aprendizaje, pues permite adquirir lo que Joel Chadabe llama conocimiento predictivo, es decir, la habilidad de intuir lo que le pasará a un sonido antes de realizar una acción para modificarlo. Todos tenemos un cierto conocimiento predictivo: por ejemplo, casi todos sabemos que, si rotamos una perilla de volumen en el sentido del reloj, el sonido que sale del amplificador aumentará de intensidad. Pero una vez que entramos en la esfera de la síntesis digital de sonido las cosas se vuelven más complejas, y se hace necesario adquirir experiencia directa de manipulación y de percepción para aumentar nuestro conocimiento predictivo.
No obstante, una completa formación sobre sonido producido digitalmente requiere mucho más que conocimiento predictivo: necesitamos saber por qué nuestras manipulaciones causan los cambios perceptivos que experimentamos. Este conocimiento teórico refuerza nuestro conocimiento experimental intuitivo, y al mismo tiempo, nuestra experiencia le da sentido perceptivo a las explicaciones teóricas.

En mi opinión, Cipriani y Giri han realizado un trabajo magistral al permitir que los conocimientos experimental y teórico se refuercen entre sí. Este libro puede ser utilizado, ya sea como libro guía en un contexto académico, ya sea como un recurso para el autoaprendizaje. Adicionalmente, este libro incluye una introducción exhaustiva al procesamiento digital de señales con Max, y además, es una introducción maravillosa a los conceptos de programación en ese software.

Como verás, los conocimientos teóricos están concentrados en los capítulos T, mientras que los conocimientos prácticos y experimentales son impartidos en los capítulos P. Estos capítulos se alternan de manera paralela, como las piernas izquierda y derecha al subir una escalera, profundizando y refinando los conceptos a niveles de sofisticación cada vez más altos.

Espero que aproveches los excelentes ejemplos en Max que los autores han creado. Son esclarecedores y al mismo tiempo entretenidos, y suenan

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

lo suficientemente bien como para ser usados en el escenario. Igualmente son dignos de examinar como modelos para tus propios patches en Max, o para extenderlos de nuevas maneras. Por supuesto, unos pocos minutos de "diversión" con los ejemplos no es lo mismo que estudiarlos en función de los conceptos tratados en el libro. Este manual, de hecho, provee el lenguaje para expresar tales conceptos en términos de la teoría asociada. Conocer la teoría es esencial, pues si estás leyendo este libro, probablemente es porque te interesa ser capaz de hacer cosas mucho más complejas que girar una simple perilla de volumen.

Este es el deseo de los autores y el mío también. Buena suerte en esta nueva aventura. Aprovecho para darle las gracias a mis dos amigos italianos por haber creado un recurso tan completo para el aprendizaje sobre música digital. ¡Ojalá hubiera existido algo así cuando yo era estudiante!

David Zicarelli
Fundador de Cycling'74, casa productora de Max.

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

PREFACIO
por Alvise Vidolin

El libro de Alessandro Cipriani y Maurizio Giri, *Musica Electrónica y Diseño Sonoro*, es un sólido texto didáctico que se dirige a personas que desean entender qué es la música electrónica, a partir de la experiencia directa sobre el sonido y sobre sus técnicas de síntesis, de procesamiento y de control, que han marcado el desarrollo de esta disciplina hasta hacerla convertir, tal como decía Luciano Berio, en "parte del pensamiento musical cotidiano". En cierto sentido, es un libro contracorriente, pues no va detrás de los "efectos especiales" ya preparados que han sido determinantes en el éxito comercial de muchos instrumentos musicales electrónicos: por el contrario, le proporciona al estudiante el método, las herramientas teóricas y la praxis operativa, no sólo para obtener un "efecto" específico, sino también para inventar otros originales, realizados en función de las propias exigencias musicales.

Los autores transforman en un método didáctico la filosofía de los llamados *instrumentos musicales abiertos* que han caracterizado el nacimiento de la música electrónica. Esta tipología de instrumentos permitía un nuevo sistema de producción musical basado en elementos "componibles", mediante el cual el compositor podía crear su música de manera integral hasta la realización sonora definitiva, la cual era memorizada en cinta magnética para sus audiciones acusmáticas en concierto. Lamentablemente, las tecnologías analógicas de los pioneros de la música electrónica no permitían realizar en vivo la complejidad musical requerida por los compositores, y hubo que esperar el desarrollo de la informática en tiempo real para que se cumpliera este deseo. Uno de los programas que ha contribuido eficazmente a este desarrollo ha sido Max, implementado en el IRCAM por Miller Puckette hacia la mitad de los años 80. Desde las primeras versiones, Max ha proporcionado las herramientas para ejercer, en vivo, un control libre de cualquier dato MIDI; en las siguientes versiones, con la añadidura de MSP, el control se ha extendido a señales de audio, y mucho más recientemente, con la llegada de Jitter, también a señales de video. La característica que hace que Max sea particularmente eficaz para la realización de cualquier producción musical, ya sea en estudio o en vivo, es su concepción abierta. Esta apertura radica en el hecho de que no proporciona ambientes musicales predefinidos, que caducan inevitablemente con la llegada de nuevas modas, sino que provee los objetos para construir el ambiente musical requerido para la composición que se desea crear o ejecutar. Y no es casualidad que Cipriani y Giri hayan elegido precisamente Max como instrumento operativo para las ejercitaciones prácticas que ilustran de manera constante y exhaustiva los capítulos teóricos del libro, ampliando sus potencialidades con bibliotecas de abstracciones que se pueden descargar de la página web del libro, al igual que otros materiales útiles.

Obviamente, los músicos que están acostumbrados a elegir un nuevo instrumento electrónico con base en los resultados musicales inmediatos que este ofrece, quedarán desilusionados ante esta perspectiva, pero si tienen la paciencia de ampliar sus propios conocimientos sobre el sonido, de entrar en

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

la lógica de la programación con objetos, y de trabajar tratando de alcanzar un objetivo, en lugar de dejarse llevar por la elección de efectos sonoros ya preparados, descubrirán las infinitas potencialidades ofrecidas por los programas informáticos abiertos, de los cuales Max es un válido ejemplo.

Esta concepción abierta es un punto crucial para el aprendizaje de la música electrónica: es el método que le permite a esta disciplina introducirse con todo derecho dentro de la tradición didáctica de la música, poniéndose a la par de los métodos analíticos y de la práctica instrumental de los cursos tradicionales de composición y de instrumento. Por otra parte, el estudio de la música no puede prescindir del estudio del sonido en sus diferentes dimensiones, integrando la rica tradición acústica con los nuevos conocimientos y los diferentes métodos de la música electrónica. En otras palabras, el músico debe estudiar el sonido, pero no solo como un dato teórico abstracto; también debe asimilarlo a través de la experiencia sensible, confrontando continuamente los elementos teóricos con la experiencia perceptivo-musical.
Este libro de *Música Electrónica y Diseño Sonoro* es, precisamente, el instrumento didáctico ideal para las nuevas generaciones de músicos, ya que logra crear un perfecto equilibrio entre los conocimientos teóricos y la práctica. La obra, articulada en cuatro volúmenes, adopta un método didáctico orgánico con una concepción abierta e interactiva de la enseñanza que se demuestra eficaz, tanto para una didáctica dirigida por un docente como para el autoaprendizaje. En muchos ejemplos prácticos los autores transforman Max en un completo "laboratorio de construcción de sonidos electrónicos": inician con los primeros sonidos de la electrónica analógica, para llegar a profundizar en las principales técnicas de síntesis y de procesamiento de señales, realizando instrumentos virtuales y de interacción, programando controles gestuales para la ejecución en vivo, creando sistemas de difusión y de espacialización para la escucha. De este modo, la didáctica se vuelve interactiva, ya que el laboratorio virtual funciona en tiempo real y permite escuchar, paso a paso, el proceso de realización, verificando rigurosamente la propia creación.

En conclusión, este libro presenta todas las características para convertirse en el manual de referencia de los cursos de música electrónica, tanto de los conservatorios italianos como de otros, y continúa el camino iniciado exitosamente con Il Suono Virtuale, escrito también por Alessandro Cipriani en colaboración con el ya fallecido Riccardo Bianchini.

Alvise Vidolin
Venecia 08/09/2009

© Contemponet 2020

INTRODUCCIÓN

Este es el primer volumen (ahora actualizado a Max 8) de una colección de libros dedicada a la síntesis y el procesamiento digital de sonido. El plan de trabajo comprende además:

- un segundo volumen que trata diferentes temas, entre ellos: el audio digital, procesadores dinámicos, líneas de retraso, el protocolo MIDI, síntesis en tiempo real, Max for Live, y un primer capítulo dedicado al arte de la organización del sonido;
- posteriormente serán abordados: la reverberación y la espacialización, diferentes técnicas de modulación (como AM, RM, PM, FM, etc.), el microsonido y la síntesis granular, la programación con GEN, análisis/resíntesis y convolución, y más.

PRERREQUISITOS

En todos los volúmenes se alternan capítulos teóricos y capítulos de práctica en la computadora, que deben ser estudiados de manera entrelazada. Este primer volumen puede ser utilizado por usuarios de distintos niveles de preparación.
El nivel mínimo requerido por las personas que empiecen a estudiar el volumen 1 comprende:

- nociones básicas de teoría musical (notas, escalas, acordes, etc.)
- una competencia básica en el manejo de computadoras (saber guardar un archivo, copiar, pegar, etc.).

El texto debe ser estudiado alternando cada capítulo de teoría con su respectivo capítulo de práctica, incluidas las actividades con la computadora. Los capítulos teóricos no son sustituto de textos más profundos sobre síntesis. Por el contrario, se trata de un compendio teórico indispensable para el trabajo práctico de programación y de creación de sonidos con la computadora, y por lo tanto, es parte de un sistema didáctico orgánico. El estudio de este volumen puede ser realizado de manera autónoma o bajo la guía de un docente.

EJEMPLOS INTERACTIVOS

Todas las secciones teóricas de este libro constan de numerosos ejemplos interactivos (algunos en formato video), disponibles en el sitio web www.**********. Gracias a estos ejemplos, el lector puede entrar en contacto con el sonido y con su proceso de creación y de procesamiento, no obstante aún no se haya enfrentado a programar por su cuenta. De esta manera, el estudio de la teoría siempre está asociado a la percepción del sonido y a sus posibles transformaciones. Nuestro objetivo es hacer que la percepción y el conocimiento vayan de la mano en el estudio del diseño sonoro y de la música electrónica, y este criterio es el que prevalece en toda la obra didáctica, la cual incluye materiales en línea adicionales que serán actualizados y ampliados gradualmente.

TEORÍA y PRÁCTICA

El enfoque didáctico de este libro está basado principalmente en la interacción (imprescindible, para nosotros) entre teoría y práctica. De hecho, uno de los problemas en el campo del procesamiento de señales, es que normalmente

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

los expertos en teoría no están implicados en la resolución de problemas concretos relacionados con la praxis de la creación de sonido; por otra parte, nos encontramos con personas (en una cantidad notoriamente mayor) que aman inventar y modificar sonidos usando su computadora, pero que a menudo tienen un escaso conocimiento técnico-teórico de dicha actividad y carecen de las competencias necesarias para modificar lo que el software en uso, dentro de sus confines rígidos, establece que se haga. Cada vez más, el mercado ofrece productos tecnológicos maravillosos, pero muy difíciles de personalizar. La información que proporcionan, a menudo, es aproximada y poco sistemática, y este factor, sumado a la rápida obsolescencia de los sistemas, contribuye a mantener a los usuarios en un estado de ignorancia, que aunque pueda parecer placentero, en cierto sentido los reduce a usar las computadoras y los programas que compran, de manera superficial, a actualizarlos continuamente sin haber comprendido la verdadera esencia de los mismos. Es por esto que afirmamos que el estudio de este manual ayudará a nuestros usuarios a adquirir un entendimiento más profundo sobre el uso de programas comerciales para la síntesis y el procesamiento de señales.

MAX

La parte práctica de este libro está basada en el software Max. Este programa, escrito originalmente por Miller Puckette, ha sido desarrollado y extendido por David Zicarelli y es publicado como un producto de Cycling '74 (www.cycling74.com). Max es un ambiente gráfico interactivo para música, procesamiento de audio y multimedia. Es usado en todo el mundo por músicos, compositores, diseñadores sonoros, artistas multimedia, etc. y se ha convertido en el estándar para proyectos creativos modernos tecnológicos, tanto en las esferas musicales como visuales.

Es un lenguaje de programación gráfico, y por lo tanto, es relativamente fácil de aprender, no obstante su gran potencialidad.

En Max es posible crear programas por medio de objetos gráficos que se conectan entre sí con cables virtuales. Estos objetos pueden realizar cálculos, producir o procesar sonidos, renderizar visuales o ser configurados como interfaz gráfica de usuario. Usando sus capacidades de síntesis de sonido y de procesamiento de señales, es posible elaborar sintetizadores, samplers, reverbs, efectos de procesamiento de señal, entre otros.

En la práctica, Max adopta la metáfora del sintetizador modular: cada módulo se encarga de una función en particular, intercambiando datos con los módulos a los que está conectado. La diferencia entre un sintetizador modular tradicional y Max es que con Max, se puede acceder y controlar un nivel de detalle que sería inconcebible en un sintetizador preconfigurado (tanto hardware como software).

ENFOQUE EDUCATIVO Y MÉTODO DE ESTE LIBRO

Sobre la base de los conceptos ya descritos, en esta obra hemos tratado de llenar el vacío de información concerniente a esta materia, continuando en la dirección ya empezada por Cipriani y Bianchini con su libro "Il Suono Virtuale",

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

dedicado a síntesis de sonido y procesamiento de señales. Las innovaciones en este nuevo texto son sustanciales, ya sea por la calidad de los ejemplos provistos como por el enfoque educativo completamente distinto. Existe muy poca disponibilidad de bibliografía sobre métodos de enseñanza de música electrónica. Es por esto que nos hemos aproximado al problema considerando varias maneras prometedoras de sondear las profundidades de la materia en cuestión. Este ejercicio nos ha llevado a la creación de un método educativo integral, en el que adoptamos varias ideas y técnicas de la didáctica de las lenguas extranjeras, para así desarrollar un concepto de enseñanza y aprendizaje más abierto e interactivo.

Además de los ejemplos interactivos, hemos incluido "agendas de aprendizaje" que detallan objetivos específicos para cada capítulo, y que incluyen actividades de escucha y análisis, evaluaciones, glosarios y sugerencias discográficas. Los capítulos prácticos del libro también incluyen muchas actividades innovadoras, a saber, el reemplazo de partes de algoritmos, la corrección, complementación y análisis de algoritmos, la construcción desde cero de nuevos algoritmos, ejercicios de ingeniería en reversa (donde el lector escucha un sonido y luego trata de inventar un algoritmo para crear un sonido similar).

Estas actividades y tareas tienen como intención activar el conocimiento y las habilidades prácticas del lector. Cuando aprendemos una lengua extranjera, existe una brecha entre lo que sabemos y lo que somos capaces de usar en la práctica. Es común que el vocabulario pasivo del estudiante (el número total de términos que el estudiante puede reconocer) sea mucho mayor que el vocabulario activo que puede realmente usar al hablar y escribir. Ocurre lo mismo para un lenguaje de programación: un estudiante puede entender cómo funcionan los algoritmos mas no ser capaz de escribirlos desde cero.
Las actividades en este libro que se concentran en reemplazar partes de algoritmos, corregir algoritmos defectuosos, y la aplicación de ingeniería en reversa, han sido incluidos para proponer problemas con los que el lector se anime a encontrar sus propias soluciones, haciendo que el proceso de aprendizaje sea más activo y creativo. Es igual cuando aprendemos un idioma extranjero realizando ejercicios de sustitución (por ejemplo, "reemplaza el verbo subrayado en la siguiente frase: me gustaría salir"), de corrección (por ejemplo, "corrige la siguiente frase: me quiero fui a casa"), y frases a completar (por ejemplo: "me gustaría ... a casa"). En este contexto, es de vital importancia que el estudiante trabaje en estas actividades para evitar una aproximación excesivamente pasiva al aprendizaje. Nuestra aproximación, de hecho, no solo involucra la interacción entre la **percepción** de los sonidos y el **conocimiento** derivado de leer el libro, sino también la interacción entre estos dos factores y las **habilidades**, **competencias** y **creatividad** propias del usuario.

Este método no está basado en una progresión rígidamente lineal; es más bien una red que le permite al usuario adquirir conocimiento y habilidades prácticas a través de una interacción en cuatro dimensiones distintas: aprendizaje de conceptos teóricos, aprendizaje del programa MaxMSP, interacción con material de ejemplo y construcción de algoritmos.

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

La obra, compuesta por varios volúmenes y una sección on-line, es una multi-plataforma, y la teoría ha sido concebida de manera tal que pueda servir de base para otros posibles textos de práctica basados en el uso de software diferentes, utilizando la misma metodología didáctica.

TIEMPO DE DEDICACIÓN

El tiempo de dedicación, obviamente, es diferente para cada persona. En particular y solo a título informativo, hacemos referencia al tiempo necesario para cada una de las dos modalidades con que se puede abordar el estudio de este manual: el autoaprendizaje y el aprendizaje bajo la guía de un docente experto.

Autoaprendizaje (300 horas de estudio individual)

Capítulos	Argumento	Total de horas
1T+1P+IA	Síntesis de sonido	100
2T+2P	Síntesis aditiva	60
3T+3P+IB	Sustractiva y filtros	110
4T+4P	Señales de Control	30

Aprendizaje con un docente
(Curso de 60 horas frontales + 120 de estudio individual)

Capítulos	Argumento	Clases	"Feedback"	Estudio	Total
1T+1P+IA	Síntesis de sonido	16	4	40	60
2T+2P	Síntesis aditiva	10	2	24	36
3T+3P+IB	Sustractiva y filtros	18	4	44	66
4T+4P	Señales de Control	5	1	12	18

INFORMACIÓN PRÁCTICA

Muchos materiales indispensables acompañan este libro para un mejor aprendizaje, entre ellos, ejemplos interactivos, *patches* (programas escritos en Max), archivos de sonido, bibliotecas de programación y otros materiales de apoyo, que se pueden descargar desde el sitio **www.**************************.

Ejemplos interactivos

Es importante realizar los ejemplos interactivos durante el estudio de un capítulo teórico, antes de continuar con el capítulo práctico. El trabajo con estos ejemplos interactivos ayudará a la asimilación de la parte práctica relativa a los conceptos discutidos en la teoría.

Archivos de ejemplo

Los archivos de ejemplo (*patches*) han sido creados para ser utilizados con Max; este software se puede descargar desde el sitio web oficial de Cycling '74: www.cycling74.com.

Alternando teoría y práctica

En este libro, los capítulos teóricos se alternan con capítulos enfocados hacia la práctica de la programación. Por esta razón, el lector tendrá que aprender

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

primero toda la teoría de un capítulo dado antes de pasar al correspondiente capítulo práctico (por ejemplo, primero todo el capítulo 1T y luego todo el capítulo 1P). Como alternativa a este enfoque, se puede optar por leer una sola sección de la parte teórica y luego ir directamente a la sección correspondiente del capítulo práctico (ejemplo: 1.1T y 1.1P, luego 1.2T y 1.2P, etc.).

Los interludios
Nótese que entre el primer y el segundo capítulo, y entre el tercero y el cuarto, hay 2 "interludios" técnicos: el Interludio A y el Interludio B, respectivamente, dedicados específicamente al lenguaje Max; estos capítulos no están relacionados con temas tratados en la parte teórica: de todas maneras de todas maneras, son muy necesarios para poder seguir el código del libro. Después de haber estudiado la teoría y la práctica del primer capítulo, y antes de continuar con el segundo, es recomendable estudiar el Interludio A. De la misma forma, el Interludio B debería ser estudiado después de haber terminado los capítulos 3T y 3P.

Aprender Max
Aprender Max (y en general, aprender síntesis y procesamiento de señales) requiere esfuerzo y concentración. En contraste con otros programas comerciales de música, Max permite mucha flexibilidad en la programación, y por lo tanto, los programadores de algoritmos gozan de una gran libertad, pero para poder aprovechar esta libertad, es aconsejable considerar las recomendaciones de este libro y programar de forma sistemática. Un aprendizaje "intuitivo" que no respete el esquema de estudio trazado en este libro, no dará resultados satisfactorios en Max, especialmente para quienes apenas estén empezando. Este software es un verdadero instrumento musical, y hay que estudiarlo como si se tratara de un instrumento tradicional (como el violín, por ejemplo), es decir, con continuidad, partiendo de lo básico para ir enfrentándose poco a poco a las técnicas más complejas; solo de esta manera es posible retener los conocimientos y las técnicas ya adquiridas, y se puede llegar a dominar el programa.

Bibliografía
Se tomó la decisión de limitar la bibliografía a una lista de trabajos de referencia absolutamente esenciales y, por supuesto, una lista de libros y artículos citados en el texto. Una bibliografía más detallada está disponible on-line.

Antes de empezar
Para empezar a trabajar con este libro, necesitarás descargar en tu computadora los ejemplos interactivos o videos relacionados, que encontrarás en la página de soporte **www.******************. Durante la lectura de los capítulos teóricos encontrarás constantes referencias a los ejemplos contenidos en esta colección descargable.

Para trabajar interactivamente con los capítulos de programación de este libro, necesitarás descargar en tu computadora la biblioteca *Virtual Sound Macros* desde la página ya mencionada. También será necesario instalar Max en tu computadora, disponible en el sitio web de Cycling '74: www.cycling74.com.

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

La página web **www.********** contiene instrucciones detalladas para una correcta instalación de Max y de la macro biblioteca *Virtual Sound Macros*; es aconsejable leer el documento titulado "Cómo instalar y configurar Max". Revisa siempre en la página de soporte los patches (programas en Max) relacionados con los ejemplos prácticos de este libro, así como también los archivos de audio para los ejercicios de Ingeniería en reversa.

Comentarios y sugerencias

Las correcciones y los comentarios son siempre bienvenidos. Favor contactar a los autores por correo electrónico a:
a.cipriani@edisonstudio.it y maurizio@giri.it
http://www.edisonstudio.it/alessandro-cipriani
http://www.giri.it

AGRADECIMIENTOS

Queremos agradecer a Gabriele Cappellani, Salvatore Mudanò y Francesco "Franz" Rosati por su paciencia y largas horas de trabajo; a Eugenio Giordani, Giuseppe Emanuele Rapisarda, Fausto Sebastiani y a Alvise Vidolin por su generosidad.

DEDICATORIAS

Este texto está dedicado a Riccardo Bianchini, quien hubiera querido participar en la realización de este texto educativo. Desafortunadamente, falleció de manera inesperada antes del inicio de este trabajo. Hemos recolectado, revisado y citado algunos de sus materiales (algunos inéditos) en algunas secciones de teoría. Esta ha sido una manera de tener a Riccardo con nosotros. Gracias especiales a Ambretta Bianchini por su gran generosidad y sensibilidad durante estos años de trabajo.

¡Feliz lectura!

Alessandro Cipriani y Maurizio Giri

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

1T
INTRODUCCIÓN A LA SÍNTESIS DE SONIDO

1.1 SÍNTESIS DE SONIDO Y PROCESAMIENTO DE SEÑALES
1.2 FRECUENCIA, AMPLITUD Y FORMA DE ONDA
1.3 VARIACIONES DE FRECUENCIA Y AMPLITUD EN EL TIEMPO: ENVOLVENTES Y GLISANDOS
1.4 RELACIÓN ENTRE FRECUENCIA E INTERVALO MUSICAL
1.5 NOTAS SOBRE EL TRABAJO CON SONIDOS SINTETIZADOS
1.6 NOTAS SOBRE PANNING

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

AGENDA DE APRENDIZAJE

PRERREQUISITOS DEL CAPÍTULO

- Conocimientos fundamentales de herramientas informáticas (operaciones básicas, manejo de carpetas, tarjetas de sonido, etc.)
- Conocimientos mínimos de teoría musical (semitonos, octavas, tiempos, etc.)

OBJETIVOS

Conocimientos

- Conocer los procesos mediante los cuales se realiza la síntesis y el procesamiento de señales
- Conocer los parámetros principales del sonido y sus características
- Conocer la codificación de la altura y de la intensidad
- Conocer la relación entre los intervalos musicales en los diferentes sistemas de afinación
- Conocer los diferentes formatos de archivo audio

Habilidades

- Saber identificar auditivamente cambios de frecuencia y de amplitud, describiendo sus características
- Saber identificar las diferentes fases de la envolvente de un sonido o de un glisando

CONTENIDOS

- Síntesis y procesamiento de señales en la computadora
- Timbre, altitud e intensidad del sonido: teoría
- Glisando y envolvente de amplitud: teoría
- Relación entre frecuencias, altitud y codificaciones midi
- Uso de sonidos muestreados. (Breves notas)

TIEMPO DE DEDICACIÓN - Capítulo 1 (Teoría y Práctica) - Interludio A

Autodidactas

Sobre un total de 300 horas de estudio individual (volumen I, teoría y práctica):

- Aproximadamente 100 horas

Cursos

Sobre un total de 60 horas de clase + 120 de estudio individual (volumen I, teoría y práctica):

- Aproximadamente 16 horas de clase frontal + 4 de "feedback"
- Aproximadamente 40 horas de estudio individual

ACTIVIDADES

- Ejemplos interactivos

EVALUACIÓN

- Prueba de preguntas de respuestas cortas
- Prueba de escucha y análisis

MATERIALES DE APOYO

- Conceptos fundamentales - Glosario

© Contemponet 2020

1.1 SÍNTESIS DE SONIDO Y PROCESAMIENTO DE SEÑALES

El uso de la electrónica y, sobre todo, de la computadora en la música, ha permitido que compositores y músicos manejen y manipulen sonido con una precisión y una libertad impensables solo con instrumentos acústicos.
Gracias a la computadora, ahora es posible modelar el sonido de cualquier forma imaginable. Se dice a menudo que, mientras el compositor "tradicional" compone usando sonidos, el compositor electrónico compone los sonidos, es decir, entra en el sonido, en sus componentes elementales, creándolos y transformándolos a su gusto.

Podríamos decir que ocurre lo mismo con la animación gráfica: gracias a la computadora es posible crear imágenes y secuencias filmadas extremadamente realistas, que serían difíciles de producir por otros medios. En la actualidad, casi todos los efectos especiales cinematográficos son realizados con computadoras, y cada vez es más frecuente encontrar entidades virtuales compartiendo la pantalla con actores de carne y hueso.

El "secreto" de esta flexibilidad radica en el paso del mundo analógico (el de los objetos concretos) al digital (el de los números): el proceso de digitalización consiste, precisamente, en transformar una información (un texto, un sonido, una imagen) en números[1]. Una vez que una imagen o un sonido han sido convertidos a una secuencia numérica, pueden sufrir cualquier tipo de transformación, ya que los números, gracias a siglos de desarrollo de las técnicas matemáticas, pueden ser transformados y manipulados de cualquier modo.

Este texto se concentrará básicamente en dos aspectos: la síntesis de sonido y el procesamiento de señales.

- La **síntesis de sonido** (sound synthesis) se refiere a la generación electrónica de sonido. Prácticamente, se trata de la posibilidad de crear sonido a partir de algunos parámetros que han sido elegidos en función del resultado sonoro que se desea obtener.

- El **procesamiento de sonidos o señales** (signal processing) se refiere a los procesos utilizados para modificar un sonido pregrabado (un sonido de guitarra grabado con anterioridad, por ejemplo), un sonido generado por una particular técnica de síntesis, etc.

[1] Ahondaremos en este concepto a lo largo del capítulo.

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

SÍNTESIS DIGITAL DE SONIDO

Para generar cualquier tipo de sonido usando un lenguaje de programación diseñado para síntesis de sonido y procesamiento de señales, escribiremos en la computadora los parámetros relativos al tipo de "máquina virtual" que queremos construir (esto es, realizaremos un **algoritmo**[2]) y las operaciones que esta máquina debe ejecutar para crear sonido.

Una vez que hayamos escrito estas instrucciones, el lenguaje de programación que estemos usando (Max por ejemplo) las ejecutará, creando, de esta forma, un flujo de datos numéricos en el que todas las características del sonido o sonidos que hemos especificado antes serán representadas digitalmente[3]. Entre la generación de este flujo de datos digitales y el momento en que escuchamos el sonido ocurre otra operación fundamental para la cual se requiere una **tarjeta de sonido**. La tarjeta lee los datos digitales y los transforma en una señal eléctrica que es alimentada a un amplificador y luego a los parlantes. En este caso, la tarjeta realiza una conversión de digital a análogo (abreviado como D/A), permitiéndonos escuchar sonidos cuyas características están especificadas en el flujo de datos digitales (fig. 1.1).

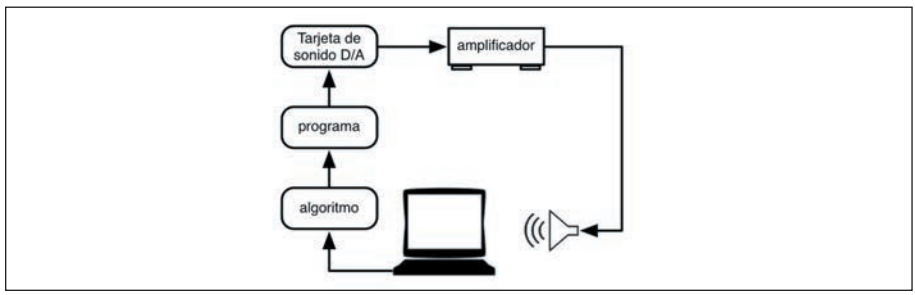


fig. 1.1: síntesis en tiempo real

Estos datos pueden ser almacenados en nuestro disco duro como un archivo de audio, lo qual nos permitirá escuchar el resultado de nuestro proceso algorítmico tantas veces como queramos, o incluso procesarlo. Cuando el flujo de datos es enviado directamente a la tarjeta de sonido mientras es procesado, se habla de síntesis en **tiempo real** (real time). Por el contrario, cuando el procesamiento de datos se realiza por completo, primero se almacena en un archivo de audio y solo posteriormente es enviado a la tarjeta de sonido para su escucha; en tal caso se habla de síntesis en **tiempo diferido** (no real time u offline). fig. 1.2.

---

[2] Un algoritmo es un procedimiento que implica una serie ordenada de instrucciones elementales: estas instrucciones, ejecutadas en sucesión, permiten resolver un problema u obtener un resultado. De manera informal podríamos decir, por ejemplo, que una receta de cocina también es un algoritmo: en efecto, se trata de una serie de instrucciones que dan como resultado un plato de comida. En informática, un algoritmo es una secuencia de instrucciones, escrita en un particular lenguaje de programación, que le permite a la computadora resolver una tarea determinada.

[3] O sea, en forma de números.

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

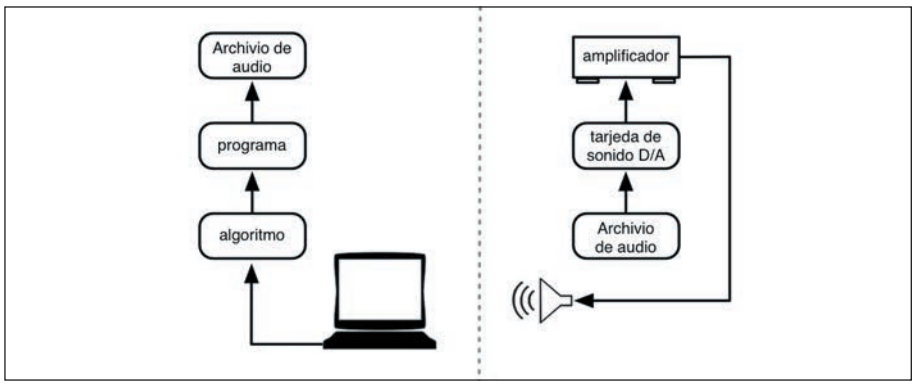


fig. 1.2: síntesis en tiempo diferido y posterior escucha

PROCESAMIENTO DE SEÑALES

El procesamiento de señales es el acto de modificar un sonido producido por una fuente sonora en vivo, o desde un archivo de audio pregrabado. Es posible hacerlo de diferentes maneras, tanto en tiempo real como en diferido. Veamos tres posibilidades:

1) SONIDO PREGRABADO EN TIEMPO DIFERIDO
PROCESAMIENTO EN TIEMPO DIFERIDO

Un sonido de flauta, por ejemplo, puede ser grabado (con un micrófono conectado a la tarjeta de sonido, que realizará una conversión análogo – digital[4]) en un archivo de audio. Podemos crear un algoritmo en el que especificaremos de qué manera debe ser modificado este sonido; posteriormente, el programa ejecutará las instrucciones y creará un nuevo archivo de audio que contendrá el sonido de flauta, procesado por la computadora según nuestras indicaciones. Por último, podremos escuchar este nuevo archivo sonoro efectuando una conversión digital – análoga (fig. 1.3)

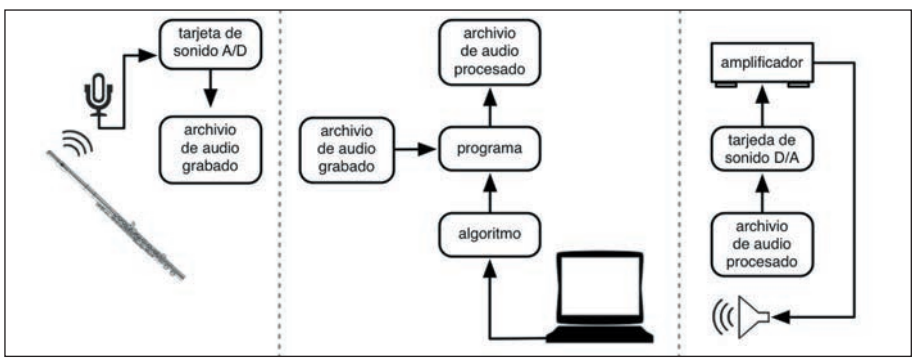


fig. 1.3: ejemplo de procesamiento de señal en tiempo diferido

[4] Es decir, que transformará un sonido real en una secuencia de números.

© Contemponet 2020

2) SONIDO PREGRABADO EN TIEMPO DIFERIDO
PROCESAMIENTO EN TIEMPO REAL

Al igual que en el ejemplo 1, el sonido es reproducido desde un archivo de audio. El programa de procesamiento, después de haber ejecutado las instrucciones, envía directamente a la tarjeta de sonido el flujo de datos que contienen el sonido procesado para escucharlo en tiempo real. Así mismo, el programa puede grabar, también en tiempo real, el sonido resultante en un archivo de audio (fig. 1.4).

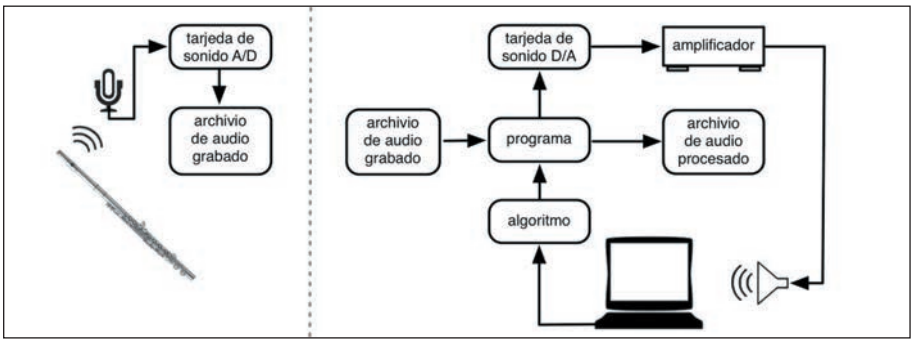


fig. 1.4: ejemplo de procesamiento en tiempo real a partir de un sonido pregrabado

3) SONIDO EN TIEMPO REAL
PROCESAMIENTO EN TIEMPO REAL

El sonido proviene de una fuente en vivo. Como en el ejemplo anterior, el programa de procesamiento, después de haber ejecutado las instrucciones, envía directamente a la tarjeta de sonido el flujo de datos que contienen el sonido procesado.

Naturalmente, también en este caso el programa puede grabar en tiempo real el sonido procesado como un archivo de audio (fig. 1.5).

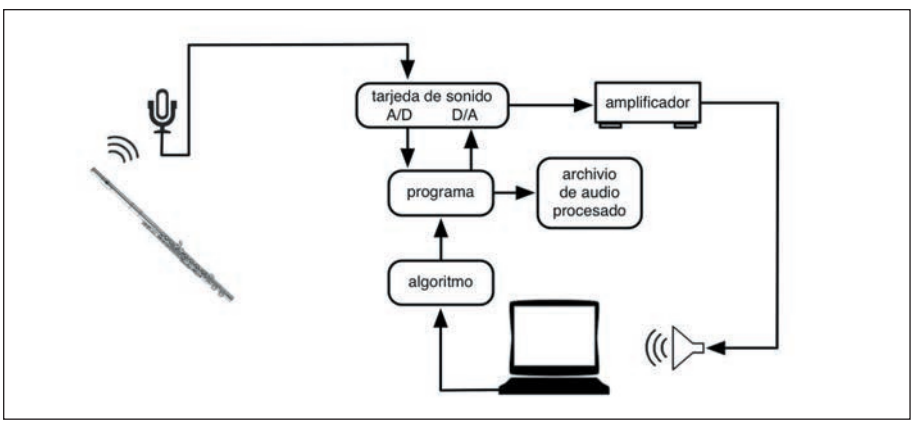


fig. 1.5: ejemplo de procesamiento en tiempo real

© Contemponet 2020

El **sistema DSP** se define como el conjunto de recursos hardware y software (tarjeta de sonido, lenguaje de programación, etc.) que permite procesar y/o sintetizar digitalmente un sonido (o señal). El término **DSP** es el acrónimo de Digital Signal Processing (Procesamiento Digital de Señales).

TIEMPO REAL – TIEMPO DIFERIDO

Hemos visto que tanto la síntesis como el procesamiento de sonido pueden llevarse a cabo en tiempo real o en diferido. A primera vista, el método más ventajoso resulta ser, sin duda alguna, el tiempo real, que nos proporciona una retroalimentación instantánea y nos permite considerar de inmediato la eficacia del algoritmo que se está implementando, al cual se le pueden hacer las modificaciones y mejoras pertinentes.

¿Para qué sirve entonces el tiempo diferido?

- Ante todo, para realizar algoritmos que la computadora no es capaz de ejecutar en tiempo real: si por ejemplo, para sintetizar o procesar un sonido que dura 1 minuto la computadora emplea 2 minutos, necesariamente habrá que grabar el resultado en el disco duro para poder escucharlo una vez que la síntesis o el procesamiento del sonido hayan terminado.

En los albores de la informática musical, todos los procedimientos de síntesis y procesamiento de sonido eran realizados en tiempo diferido, puesto que las computadoras de entonces no tenían suficiente potencia en tiempo real. Aumentando la potencia de las computadoras, fue posible realizar algunos procesos directamente en tiempo real, y con el pasar del tiempo, la capacidad de una computadora personal de realizar algoritmos de síntesis y procesamiento del sonido en tiempo real ha aumentado enormemente. Naturalmente, por más potentes que puedan llegar a ser las computadoras, siempre será posible imaginar un proceso tan complejo que requiera el uso del tiempo diferido.

- Existe también una segunda categoría de procesos que, conceptualmente, se realizan en tiempo diferido, independientemente de la potencia de cálculo del procesador: supongamos por ejemplo que queremos realizar un algoritmo que, dada una secuencia musical ejecutada por un instrumento, descomponga tal secuencia en notas y luego las vuelva a ordenar de la más grave a la más aguda.

Para implementar este algoritmo necesitamos la secuencia completa que, probablemente, estará grabada en un archivo de audio, de manera que la computadora pueda analizarla en su totalidad, identificar la nota más grave y sucesivamente las otras.
Obviamente, este análisis puede realizarse únicamente en tiempo diferido, después de la ejecución: la única computadora que podría implementar este algoritmo en tiempo real (es decir, mientras el instrumento está sonando) sería una computadora ¡capaz de prever el futuro!

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

- Otro motivo por el cual se recurre al tiempo diferido es para ahorrar tiempo. Contrariamente a lo que se puede pensar, el tiempo real no corresponde a la máxima velocidad de procesamiento posible. Imaginemos, por ejemplo, que debemos modificar un archivo de sonido de 10 minutos de duración con una particular técnica de procesamiento: si este se lleva a cabo en tiempo real, obviamente, la computadora tardará 10 minutos en ejecutarlo. Pero supongamos que nuestra computadora es lo suficientemente potente como para realizar este procesamiento en un minuto, usando el tiempo diferido. Esto significa que para aquella particular técnica de procesamiento, la computadora puede ejecutar cálculos a una velocidad 10 veces superior al tiempo real, por consiguiente, será conveniente recurrir al tiempo diferido.

1.2 FRECUENCIA, AMPLITUD Y FORMA DE ONDA

La frecuencia, la amplitud y la forma de onda son tres parámetros fundamentales del sonido. Cada uno de estos parámetros influye en la percepción sonora del oyente, en particular:

a) en la posibilidad de distinguir un sonido grave de uno agudo (frecuencia);
b) en la posibilidad de distinguir un sonido de intensidad fuerte de uno de intensidad menor (amplitud);
c) en la posibilidad de distinguir timbres diferentes (forma de onda)[5].

Veamos una tabla (Bianchini, R., 2003) que muestra la correspondencia existente entre: características físicas del sonido, parámetros musicales y sensaciones sonoras.

CARACTERÍSTICAS	PARÁMETRO MUSICAL	SENSACIÓN
Frecuencia	Altura	Aguto ↔ Grave
Amplitud	Intensidad	Forte ↔ Piano
Forma de onda	Timbre	Claro ↔ Oscuro Armónico ↔ Inarmónico

TABLA A: correspondencia entre características del sonido, parámetros musicales y sensación sonora.

FRECUENCIA

La **frecuencia** es el parámetro físico que determina la altura de un sonido; la altura es aquella característica que permite distinguir un sonido agudo de uno grave. La gama de frecuencias audibles del ser humano se extiende, aproximadamente,

[5] Veremos más adelante cómo el parámetro del timbre depende, en realidad, de diversos factores concomitantes.

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

de 20 a 20000 Hertz, es decir, de 20 a 20000 ciclos por segundo (explicaremos de qué se trata más adelante): por debajo de la mínima frecuencia perceptible, 20 ciclos por segundo, se encuentran los infrasonidos; por encima de la frecuencia máxima, 20000 ciclos por segundo, se encuentran los ultrasonidos.[6] Si nos concentramos en el campo de las frecuencias audibles, o sea, de los sonidos, podremos afirmar que entre mayor sea la frecuencia, más agudo será el sonido.
¿Pero qué se entiende por Hertz o "ciclos por segundo"? Para responder a esta pregunta, leamos la definición de sonido que nos da Riccardo Bianchini:

"El sonido es un fenómeno mecánico producido por una perturbación de un medio de transmisión (generalmente el aire) que posee ciertas características para ser percibido como tal por el oído humano.[7]
Veamos un ejemplo. La vibración es transmitida a través del aire por una cuerda vibrante (fig. 1.6).

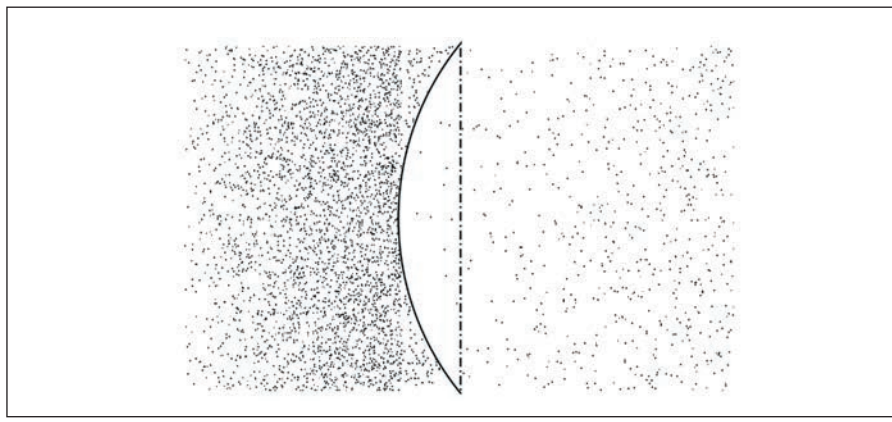


fig. 1.6: vibración de una cuerda

Esta se desplaza hacia adelante y hacia atrás, y durante este desplazamiento comprime las partículas de aire (moléculas) por un lado y las expande por el otro. Sucesivamente, el movimiento se invierte, y las moléculas que antes habían sido comprimidas se expanden, y viceversa.
Las compresiones y las expansiones (es decir, las perturbaciones del aire que inicialmente estaba en estado de quietud) se propagan después, con una cierta velocidad, a través del aire alrededor en todas las direcciones, dando lugar a ondas esféricas. Inicialmente, la densidad de las moléculas de aire es constante, o sea que en cada unidad de volumen (por ejemplo en un cm^3) hay el mismo número de moléculas.

6 En realidad, la máxima frecuencia audible disminuye con la edad.

7 Hay muchas teorías sobre la naturaleza del sonido: Roberto Casati y Jerôme Dokic sostienen que el aire es un medio a través del cual el sonido se transmite, pero que el sonido en sí es un evento localizado en el cuerpo resonante, es decir, en el sistema mecánico que produce la vibración. (Casati, R., Dokic, J. 1994). Otro punto de vista es el de Frova: "con el término «sonido» debería entenderse la sensación, así como esta se manifiesta a nivel cerebral, de una perturbación de naturaleza mecánica, de carácter oscilatorio, que interesa el medio interpuesto entre la fuente y el oyente" (Frova, A., 1999, pag. 4).

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

Esta densidad puede ser expresada por un valor de presión. Cuando el aire es perturbado, el valor de presión ya no es constante sino que varía de punto a punto: aumenta donde las moléculas son comprimidas, disminuye donde las moléculas son expandidas (fig. 1.7).

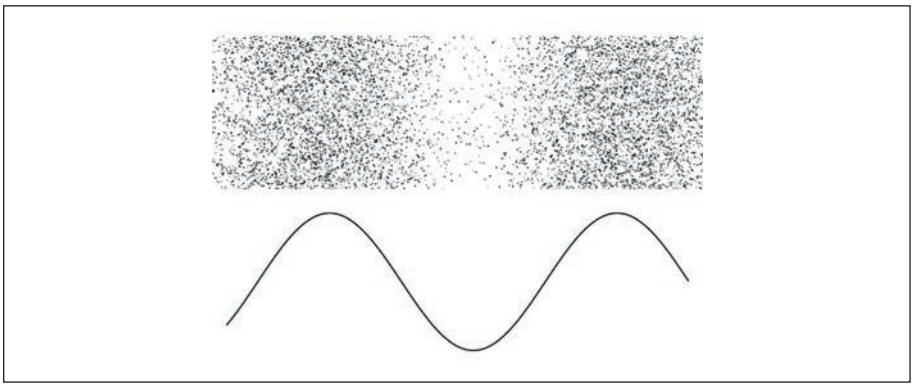


fig.1.7: compresión y rarefacción de las moléculas de aire

El fenómeno puede ser estudiado tanto desde el punto de vista del espacio (observando el valor de la presión en diferentes puntos en un determinado instante) como desde el punto de vista del tiempo (midiendo el valor de la presión en un mismo punto en función del tiempo). Por ejemplo, si imaginamos que nos encontramos en un determinado punto, sentiremos una sucesión de compresiones y expansiones del aire (fig. 1.8).

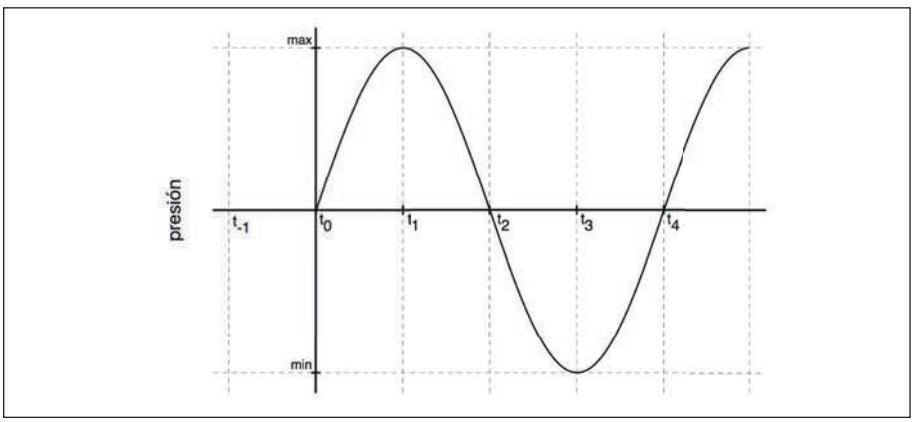


fig.1.8: representación gráfica de compresión y rarefacción

En el instante t_{-1}, o sea antes del instante t_0, la presión del aire está en su valor normal, ya que la perturbación todavía no ha llegado a nuestro punto de observación. En el instante t_0 la perturbación llega a nuestro punto de observación, inicia a crecer, llega al máximo en el instante t_1, luego decrece hasta volver al valor normal en el instante t_2, sigue decreciendo y llega al valor mínimo en el instante t_3, para después volver a subir hasta el valor normal en el instante t_4, y así sucesivamente.

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

Hasta aquí se ha descrito un **ciclo** del fenómeno. Si esto se repite siempre del mismo modo, se dice que el fenómeno es periódico.[8] El tiempo necesario para que se complete un ciclo es denominado **periodo**, se indica con el símbolo T y se mide en segundos (s) o en milisegundos (ms). El inverso del periodo, es decir, el número de ciclos que se completan en un segundo, es denominado frecuencia, y se mide en Hertz (Hz) o ciclos por segundo (cps).
Si por ejemplo una onda sonora tiene un periodo T= 0.01 s (1/100 de segundo), su frecuencia será:
1/T = 1/0.01 = 100 Hz (o 100 ciclos por segundo)" (ibidem).

Escuchemos los sonidos del ejemplo interactivo[9] número 1A, observando la figura 1.9: podemos constatar que, a mayor número de ciclos por segundo (Hz), más agudo será el sonido correspondiente.

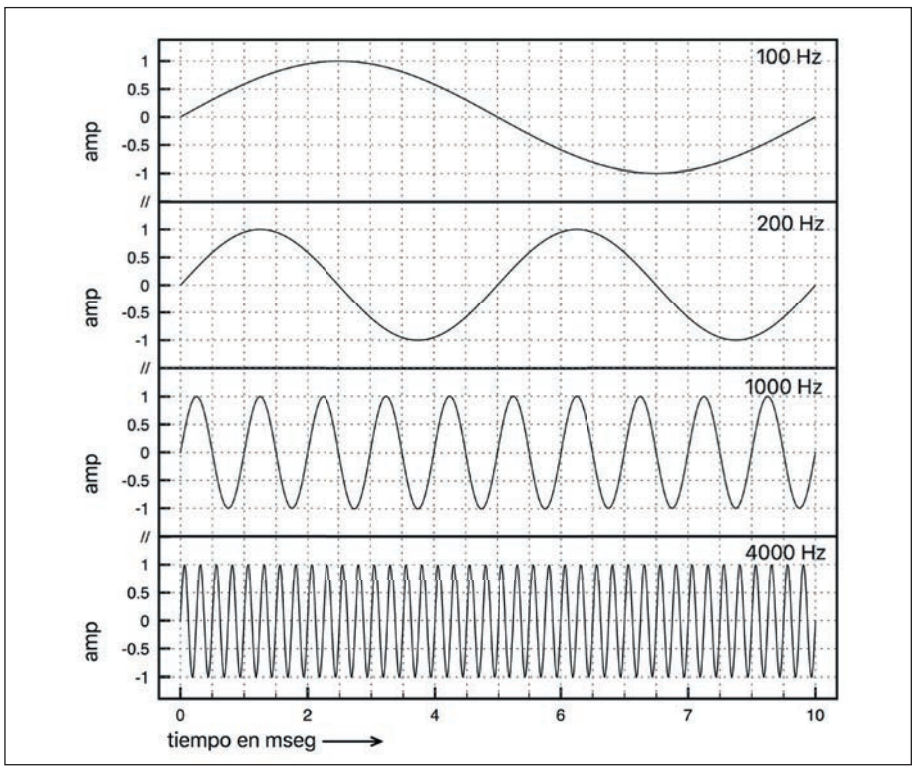


fig.1.9: cuatro sonidos de frecuencia diferente

[8] Matemáticamente, se dice que una forma de onda es periódica si se repite de manera regular y al infinito: en la práctica musical, naturalmente, nos contentamos con duraciones ¡muy inferiores! En general, una onda se considera periódica, musicalmente, cuando se repite de manera regular por un tiempo lo suficientemente largo como para generar la sensación de altura correspondiente al periodo de la onda. Ahondaremos en la cuestión en el capítulo 2.
[9] Los ejemplos interactivos y los otros materiales de apoyo del libro se encuentran en la página: http://www.**********.

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

EJEMPLO INTERACTIVO 1A • FRECUENCIA

Desde el momento en que se propaga en el espacio, una onda tiene una longitud que es inversamente proporcional a su frecuencia. Aclaremos este concepto: la velocidad del sonido en el aire (o sea, la velocidad con la cual se propagan las ondas sonoras a partir de la fuente) es de aproximadamente 344 metros por segundo.[10] Esto significa que una hipotética onda de 1 Hz tendría una longitud de aproximadamente 344 metros, ya que cuando ha completado un ciclo ha pasado un segundo, y en este tiempo se ha extendido en el espacio por una longitud de 344 metros. Una onda de 10 Hz, en cambio, en un segundo realiza 10 ciclos que se disponen en el espacio de 344 metros, ocupando cada uno 34.4 metros, es decir un décimo del espacio total. Siguiendo el mismo razonamiento, una onda de 100 Hz mide 3.44 metros: es evidente que al aumentar la frecuencia disminuye la longitud, y por lo tanto, como ya se ha dicho, las dos magnitudes son inversamente proporcionales.

AMPLITUD

El segundo parámetro fundamental del sonido es la **amplitud**, que aporta información sobre la variación de presión sonora, permitiéndonos distinguir un sonido de intensidad fuerte de uno de intensidad débil.
La presión sonora más débil que el oído humano es capaz de percibir es denominada **umbral de audición**, mientras que la presión sonora máxima que un oyente puede soportar se llama **umbral del dolor**, ya que si se sobrepasa este nivel, se producen una verdadera sensación de dolor físico y daños permanentes al órgano del oído.
Si observamos el fenómeno representado en la figura 1.10, podemos ver que el valor máximo de la presión es denominado **amplitud de pico** de la onda sonora, mientras que el valor de la presión en un punto cualquiera es llamado **amplitud instantánea**.
Cuando se indica la amplitud de un sonido, se hace referencia al valor de la amplitud de pico del mismo (fig. 1.10).
Por ejemplo, si indicamos una amplitud de pico = 1, tendremos una onda que parte de una amplitud instantánea = 0 (en el instante t_0); después, la amplitud instantánea inicia a crecer, llega al máximo en el instante t_1 (valor 1), posteriormente decrece hasta volver al valor 0 en el instante t_2, sigue decreciendo y llega al mínimo en el instante t_3 (-1), para luego volver a subir hasta el valor 0 en el instante t_4, y así sucesivamente. Esta es la representación de la amplitud de una onda sonora en función del tiempo. El proceso de digitalización transforma tal amplitud en una serie de números comprendidos entre 1 y -1.

[10] Para ser exactos, esta velocidad se alcanza cuando la temperatura es de 21°. La velocidad del sonido, de hecho, es proporcional a la temperatura del aire.

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

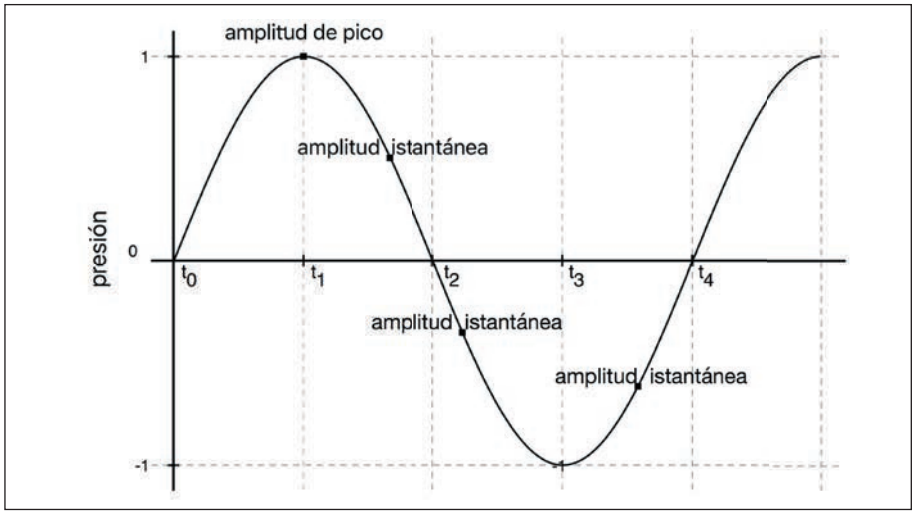


fig.1.10: amplitud de un sonido

Los valores obtenidos de este modo pueden ser usados para representar gráficamente la forma de onda (fig. 1.11). La posición en la cual se encuentra el ciclo de onda en un determinado instante es llamada **fase**.
Ahondaremos en el concepto de fase en la sección 2.1.

...

© Contemponet 2020

El capítulo continúa con:

Forma de onda
La sinusoide
Otras formas de onda
Ondas bipolares y unipolares
Uso de logaritmos en el cálculo de la presión sonora en db

1.3 VARIACIONES DE FRECUENCIA Y AMPLITUD EN EL TIEMPO: ENVOLVENTES Y GLISANDOS
Envolventes de instrumentos acústicos
Envolventes de sonidos sintetizados
Glisandos
Curvas exponenciales y logarítmicas

1.4 RELACIÓN ENTRE FRECUENCIA E INTERVALO MUSICAL

1.5 NOTAS SOBRE EL TRABAJO CON SONIDOS SINTETIZADOS
La digitalización del sonido

1.6 NOTAS SOBRE PANNING

ACTIVIDADES
• Ejemplos interactivos

EVALUACIÓN
• Prueba de preguntas de respuestas cortas
• Prueba de escucha y análisis

MATERIALES DE APOYO
• Conceptos fundamentales - Glosario

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

1P
SÍNTESIS DE SONIDO CON MAX

1.1 PRIMEROS PASOS CON MAX
1.2 FRECUENCIA, AMPLITUD Y FORMA DE ONDA
1.3 VARIACIONES DE FRECUENCIA Y AMPLITUD EN EL TIEMPO: ENVOLVENTES Y GLISANDOS
1.4 RELACIÓN ENTRE FRECUENCIA E INTERVALO MUSICAL, Y ENTRE AMPLITUD Y NIVEL DE PRESIÓN SONORA
1.5 NOTAS SOBRE EL TRABAJO CON SONIDO MUESTREADO
1.6 NOTAS SOBRE PANNING
1.7 ALGUNOS FUNDAMENTOS DE MAX

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

AGENDA DE APRENDIZAJE

PRERREQUISITOS DEL CAPÍTULO

- Conocimientos fundamentales de herramientas informáticas (operaciones básicas, manejo de carpetas, tarjetas de sonido, etc.)
- Conocimientos mínimos de teoría musical (semitonos, octavas, tiempos, etc.)
- Contenidos teóricos del capítulo 1

OBJETIVOS

Habilidades

- Saber utilizar todas las funciones básicas del software Max
- Saber sintetizar sonidos tanto en secuencia como superpuestos, usando osciladores de onda sinusoidal, cuadrada, triangular y diente de sierra
- Saber controlar de forma continua la amplitud, la frecuencia y la espacialización estéreo de un sonido (usando envolventes lineales y exponenciales para glisandos, amplitud y ubicación del sonido en la imagen estéreo)
- Saber generar secuencias aleatorias de sonido sintetizado
- Saber trabajar a nivel básico con sonido muestreado

Competencias

- Saber realizar un primer estudio sonoro de dos minutos de duración, basado en las técnicas adquiridas en este capítulo, y grabarlo como un archivo de audio

CONTENIDOS

- Síntesis de sonido y procesamiento de señales usando la computadora
- Timbre, altura e intensidad del sonido
- Glisando y envolventes de amplitud
- Relaciones entre frecuencia, altura y codificación MIDI
- Notas sobre el trabajo con sonido muestreado

TIEMPO DE DEDICACIÓN - Capítulo 1 (Teoría y Práctica) - Interludio A

AUTODIDACTAS

Sobre un total de 300 horas de estudio individual (volumen I, teoría y práctica): Aproximadamente 100 horas

CURSOS

Sobre un total de 60 horas de clase + 120 de estudio individual (volumen I, teoría y práctica): Aproximadamente 16 horas de clase frontal + 4 de "feedback" Aproximadamente 40 horas de estudio individual

ACTIVIDADES

- Sustitución de partes de algoritmos, corrección, compleción y análisis de algoritmos, construcción de nuevos algoritmos

EVALUACIÓN

- Proyecto de ingeniería en reversa

MATERIALES DE APOYO

- Lista de comandos principales de Max - Lista de objetos de Max - Lista de mensajes, atributos y parámetros de objetos de Max específicos - Glosario

© Contemponet 2020

1.1 PRIMEROS PASOS CON MAX

Para continuar con este capítulo necesitarás tener la aplicación Max correctamente instalada en tu computadora. Si no lo has hecho, puedes descargarla desde el sitio web de Cycling74 ubicado en http://www.cycling74.com. También será necesario descargar el material de apoyo para este libro que se encuentra en http://www.**********. Te recomendamos especialmente que instales la biblioteca *Virtual Sound Macros*; para ello, debes seguir las instrucciones de instalación que se encuentran en el mismo sitio web.

Iniciamos la aplicación y seleccionamos la entrada *New Patcher* desde el menú File (también es posible hacerlo digitando: Command-n, en Mac o Control-n, en Windows)[1]. Aparecerá una ventana denominada **Patcher Window** en la cual conectamos objetos de Max para crear algoritmos. Antes de seguir, observa que la *Patcher Window* está enmarcada con un borde oscuro que contiene diversos íconos (fig.1.1). Los cuatro lados de este borde reciben el nombre de *Patcher Window Toolbars*; explicaremos cada uno de los íconos gradualmente, a medida que sea necesario.

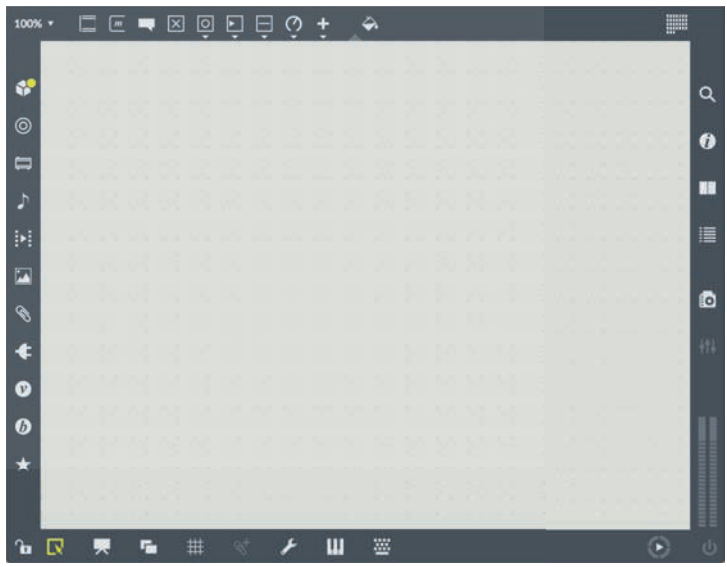


fig.1.1: *Patcher Window*

Empezaremos con la sección superior: estos íconos representan los objetos de Max que tenemos a disposición para construir "máquinas virtuales", las cuales representan algoritmos para síntesis de sonido y procesamiento de señales. De hecho, los objetos de Max pueden ser interconectados entre sí: la información fluye (bajo forma de números, señales digitales, y otros tipos de datos) de un

[1] En Mac OS X mantén presionada <Command> y luego "n", o en Windows presiona "n" mientras sigues presionando <Control>.

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

objeto a otro a través de estas conexiones. Cada objeto está diseñado para realizar una operación específica en la información que recibe, y pasará sus resultados a los objetos a los cuales esté conectado.
Una colección de objetos conectados entre sí recibe el nombre de **patch** (en referencia al antiguo mundo de los sintetizadores modulares análogos que eran programados usando cables físicos para hacer conexiones, llamados **patch cords**).

Vamos a crear nuestro primer *patch*. Si haces click en el primer ícono a la izquierda de la barra superior, aparecerá un objeto gráfico en la *Patcher Window* (ver fig. 1.2).

fig.1.2: *object box*

Este es el objeto genérico de Max y recibe el nombre de **object box** (caja de objeto). Es el objeto que usaremos más a menudo, y su función depende del nombre que le damos cuando escribimos un *string* [2] en su interior.
Para empezar, vamos a ver cómo se crea un oscilador de onda sinusoidal. Escribe la palabra "`cycle~`" en el interior de la *object box*. Observa que apenas empiezas a escribir, aparece un menú que muestra una lista de todos los objetos cuyo nombre o cuya descripción contiene los caracteres que has escrito. Esta es una función muy útil llamada **auto-compleción** (ver fig. 1.3).

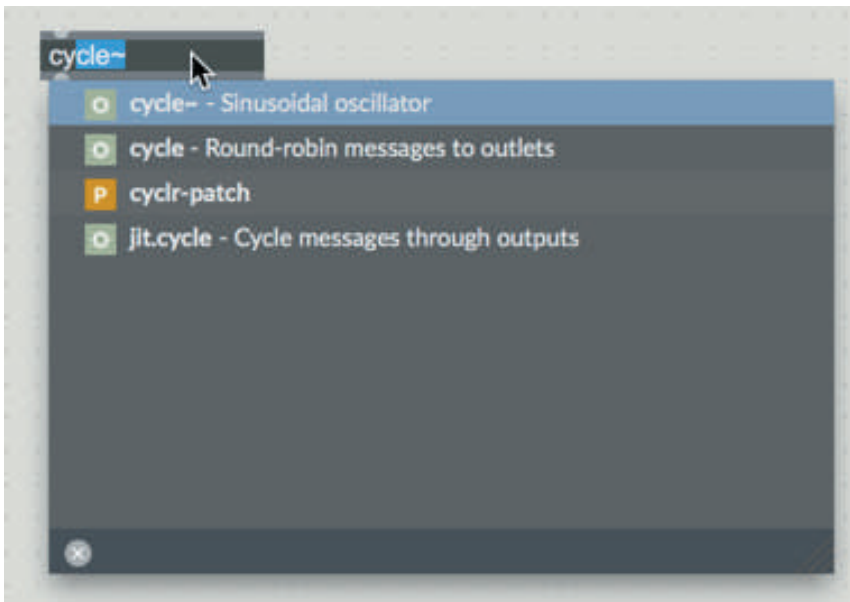


fig.1.3: menú de auto-compleción

[2] La palabra "*string*" es jerga de programación de computadoras que se refiere a una combinación de letras y números como "print", "save", "unstringquecontieneelnumero1" o "subseccion22".

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

En esta figura podemos observar que el menú de *auto-compleción* aparece tras haber escrito en el interior de la *object box* los dos primeros caracteres de "`cycle~`" ("`c`" e "`y`"). La letra que precede a cada opción del menú de auto-completión especifica el tipo de elemento sugerido por esta función. En la figura, por ejemplo, podemos ver dos letras: la "`o`" indica que se trata de un objeto, mientras que la "`p`" que indica que se trata de una abstracción. Más adelante hablaremos de las abstracciones y de otros tipos de elementos que pueden ser encontrados en un *patch* de Max. A medida que vamos escribiendo nuevas letras en el interior de una *object box*, es posible ver el nombre completo del objeto con mayor probabilidad de ser usado (a menudo, un nombre que ya has escrito antes). Puedes seleccionar el nombre del objeto que estás buscando haciendo click en el ítem respectivo, o simplemente, escribiendo el resto del nombre; para propósitos de este manual, asegúrate de seleccionar "`cycle~`" y no "cycle" [3] Ahora teclea un espacio después de "`cycle~`". El menú de auto-completión mostrará dos nuevas categorías: "Arguments" y "Attributes" (argumentos y atributos, respectivamente). Sin entrar en detalles, digamos que los elementos que pueblan estas categorías representan palabras que pueden ser escritas, opcionalmente, a continuación del nombre del objeto. Pero ignora por ahora estas sugerencias y añade, después del espacio, el número 440 (¡este espacio es muy importante!). Luego haz click en una parte vacía de la *Patcher Window* [4] verás que ahora la *object box* se parece a la de la figura 1.4

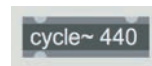


fig.1.4: objeto `cycle~`

Las zonas un poco más claras que aparecerán en los bordes superior e inferior del objeto son las entradas (**inlet**) y salidas (**outlet**), que examinaremos dentro de poco. N.B.: si el objeto no se ve como el de la figura 1.4, significa que hay un problema; lee la parte correspondiente a las preguntas frecuentes (FAQ) al final de esta sección.

[3] Observa el carácter "~", llamado **tilde**, que se encuentra a continuación de la palabra cycle. Este carácter casi siempre aparece al final de los nombres de objetos usados para procesar flujos de audio digital. Es una convención de denominación importante de Max. Dado que algunos objetos externos no aparecen en el menú de autocompletar, a veces tendrás que teclear los nombres de objeto directamente en la *object box*, por lo tanto, es importante que sepas dónde ubicar la tilde en tu teclado, especialmente si el tuyo no está en inglés. De hecho, este carácter se obtiene gracias a la combinación de algunas teclas, y puede variar dependiendo del sistema operativo que se esté utilizando o de la lengua en la que esté configurado el teclado. Por ejemplo, en el teclado italiano del Macintosh, la tilde se obtiene presionando las teclas *alt-5*. En la mayoría de los teclados de PC Windows, hay que escribir *alt-126*, si se usa el teclado numérico de la derecha, de lo contrario, si se carece de este (como en las computadoras portátiles), es posible obtenerlo manteniendo presionada la tecla *fn* para activar el teclado numérico interno en el teclado de letras, y digitar *alt-126*. Si no funciona, se puede aprovechar de la función de autocompletar, digitando el nombre de un objeto cualquiera que lleve la tilde (como por ejemplo `cycle~`) y reemplazar manualmente el nombre en la *object box* (¡dejando la tilde, obvio!)

[4] De manera alternativa, puedes presionar <Enter> en Mac o <Shift-Enter> en Windows.

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

Ahora vamos a crear el objeto **`gain~`**, cuya apariencia es similar a la de los faders de un mezclador (ver fig. 1.5).

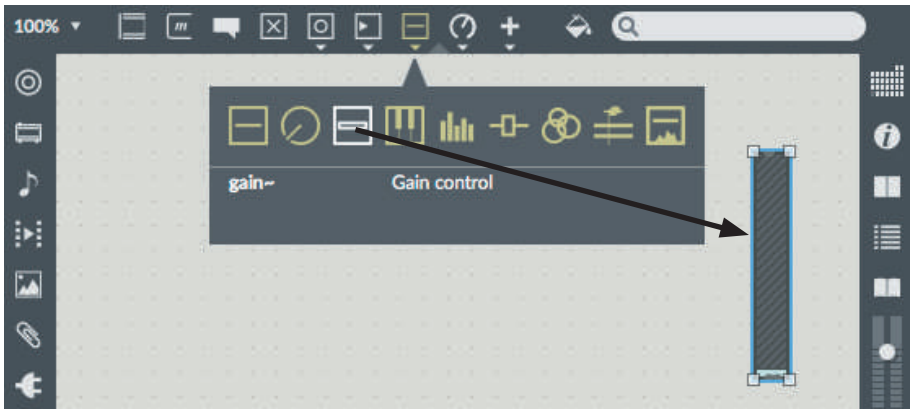


fig.1.5: objeto `gain~`

Haciendo click en el séptimo ícono de la barra superior, aparecerá una paleta que contiene todos los objetos de la categoría "Slider". Haz click en el tercer ícono de la paleta, o de forma alternativa, arrastra este ícono directamente a la *Patcher Window* usando el ratón.

Aparecerá un nuevo tipo de objeto: en lugar de una *object box*, será creado un objeto nuevo con interfaz de usuario (conocido también como objeto ui: ***user interface object***). Aquí va un pequeño truco: si te cuesta mucho encontrar un objeto en la barra de herramientas superior, simplemente crea una *object box* (el mismo objeto genérico que usamos para crear **`cycle~`**), y luego escribe en su interior el nombre del objeto que quieres crear, por ejemplo **`gain~`**. Luego haz click fuera del objeto, y este se transformará en el respectivo *ui object*.

Posiciona este objeto debajo de **`cycle~`**, y conecta la salida de este último a la entrada de **`gain~`** de la siguiente forma: acerca el puntero del ratón a la salida que se encuentra en el borde inferior del objeto **`cycle~`**; aparecerá un círculo amarillo y un recuadro (llamado *Assistance bubble*: burbuja de ayuda). El círculo (fig. 1.6a) indica que se ha seleccionado la salida; haciendo click y arrastrando con el ratón hacia abajo, aparecerá un cable con rayas negras y amarillas. Si acercas el puntero del ratón a la esquina superior izquierda del objeto **`gain~`**, aparecerá un círculo rojo con una *Assistance bubble* y un texto descriptivo (ver fig. 1.6b.); a continuación, haz click nuevamente con el ratón para realizar así la conexión entre los dos objetos.

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

a

b

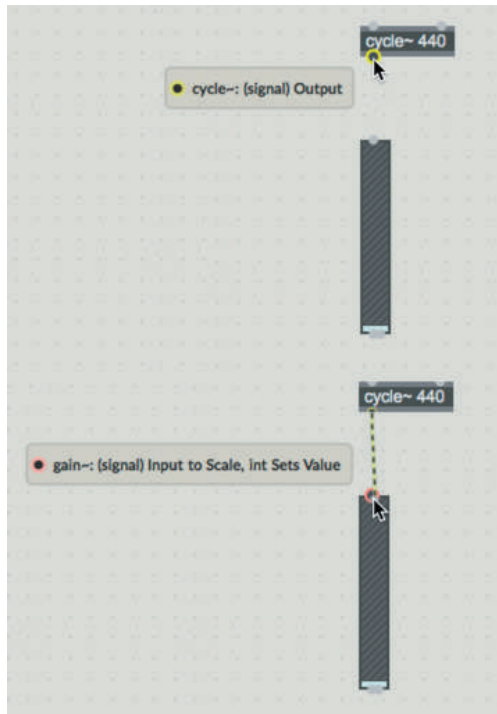


fig. 1.6: conectando dos objetos

Hemos conectado el oscilador al objeto **`gain~`**, que nos servirá para regular el volumen. Ahora debemos dirigir la señal de audio a la salida: haz click en el penúltimo ícono de la barra superior para hacer que aparezca una paleta de seis categorías de objetos[5]: selecciona la categoría "Audio" y haz click en el segundo ícono (o arrástralo dentro de la *Patcher Window*, como se muestra en la fig. 1.7). El nombre de este objeto es **`ezdac~`**.

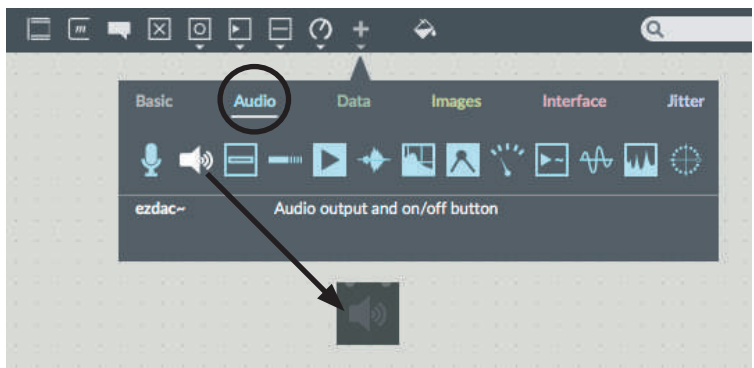


fig.1.7: objeto **`ezdac~`**

[5] Más adelante veremos cómo está organizada la barra de herramientas superior "Toolbar".

© Contemponet 2020

Ubícalo bajo el objeto **gain~** y conecta la salida izquierda de este último a las dos entradas de **ezdac~** (ver fig. 1.8).

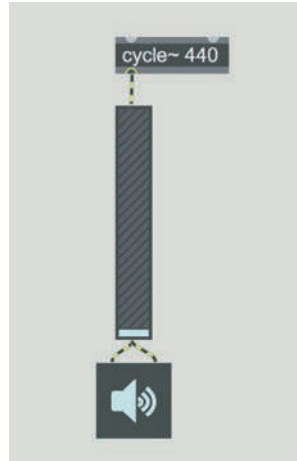


fig.1.8: nuestro primer *patch*

¡Atención! El objeto **gain~** tiene dos salidas difíciles de distinguir, por lo tanto, verifica que hayas usado la salida izquierda para hacer ambas conexiones. La mejor manera de cerciorarse de esto es leer el texto de la *Assistance Bubble* que aparece en la base de **gain~** cuando haces la conexión, y asegurarte de que contenga el texto "**gain~: (signal) Scaled Output**".

Si uno de los dos cables es gris y no con rayas amarillas y negras como aparecen en la figura 1.8, significa que, por equivocación, has usado la salida derecha; en tal caso, debes eliminar el cable seleccionándolo con un click (el cable se verá un poco más grueso) y presionando la tecla <Delete> (la misma que usas para borrar texto). El cable desaparecerá y podrás volver a conectar los objetos de manera correcta.

Ahora sería un buen momento para grabar este nuevo *patch* en el disco, pero ten en cuenta lo siguiente: ¡NUNCA grabes tus *patches* en un archivo que comparte el mismo nombre que un objeto de Max! Por ejemplo, no le pongas de nombre **cycle~** a este *patch* (¡ni tampoco cycle, sin el símbolo ~ !). Al hacer esto, Max se confunde, y puede causar resultados inesperados la primera vez que trates de volver a correr tu *patch*. Dado que es imposible recordar todos los nombres de los objetos de Max, una buena técnica para evitar el peligro del usarlos como nombres de objeto, es darle a tus archivos un nombre compuesto por más de una palabra: "oscilador de prueba", por ejemplo, o "prueba de objeto cycle~", o cualquier otra combinación. No existen objetos de Max con nombres formados por más de una palabra. ¡No olvides esta advertencia! Un gran porcentaje de los problemas encontrados por principiantes en Max están relacionados con el hecho de grabar archivos que comparten un nombre con algún objeto interno. Volveremos a hablar de este tema en el "Interludio", a continuación de este capítulo.

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

Ya hemos terminado de implementar nuestro primer *patch*, y estamos listos para ejecutarlo. Pero todavía falta un último detalle: hasta ahora hemos estado en **Edit Mode** (modo edición), es decir, la modalidad en la que podemos ensamblar *patches* insertando, moviendo y conectando objetos; vamos a hacer la transición al **Modo Ejecución** (*performance mode*), donde podremos escuchar y probar nuestro *patch*. Para ello, haz click en el pequeño candado que aparece en la esquina inferior izquierda de la *Patcher Window*, o presiona <Mac: Command-e><Win: Control-e>[6]. Una vez que entramos al modo ejecución, el candado de la esquina inferior izquierda aparece como cerrado (si aparece abierto, ¡significa que sigues en el modo edición!).
Habiendo cambiado al modo ejecución, haz click en el objeto **`ezdac~`** (el pequeño parlante), el ícono se iluminará; luego, lentamente, sube el nivel del slider **`gain~`**. Deberías de escuchar un sonido, la nota La sobre el Do central. Haciendo de nuevo click sobre el pequeño parlante, es posible deshabilitar la salida de sonido del *patch*. Si no has escuchado ningún sonido, consulta el FAQ al final de esta sección.
Ahora que hemos construido un *patch* y lo hemos visto en funcionamiento, revisemos nuestro algoritmo: el objeto **`cycle~`** es un oscilador (un generador de sonido que produce una forma de onda periódica, en este caso, una onda sinusoidal), y el número 440 que hemos escrito en su interior indica la frecuencia que queremos que produzca; en este caso, especificamos que la onda sinusoidal[7] se repita 440 veces por segundo[8].

En otras palabras, **`cycle~`** es el nombre del objeto, y 440 es el **argumento**, un valor usado por el objeto para especificar su operación. En este caso, el argumento 440 hace que **`cycle~`** produzca un sonido a 440 Hz.
La salida de señal del objeto **`cycle~`** fue conectada a la entrada del objeto **`gain~`**; esto hace que la señal generada por **`cycle~`** sea pasada al objeto **`gain~`**, donde puede ser modificada, como hemos visto, moviendo el fader de volumen. La señal modificada luego es pasada al objeto **`ezdac~`** (el pequeño parlante), cuya función es enviarla a la tarjeta de sonido, la cual administra la conversión digital - análoga de la señal, es decir, transforma la representación numérica del sonido en una señal audio que puedes escuchar en los audífonos o parlantes conectados a la computadora. De hecho, el nombre "ezdac~" es un pseudo acrónimo de *EaSy Digital to Analog Converter* (Conversor Digital-Análogo Simple).

Sigamos estudiando en detalle este *patch*. Además de escuchar el sonido, es posible "verlo". Graba el *patch* que acabas de crear en una carpeta apropiada, por ejemplo "Mis *Patches*" (la usaremos en la siguiente sección), y cierra la *Patcher Window*.
Si no has descargado y desempaquetando los archivos "Material_Capitulos_Max_Vol 1", que encuentras en http://www.**********, hazlo ahora.

---

[6] Alternativamente, puedes moverte atrás y adelante entre los modos de edición y ejecución presionando la tecla <Command> (en Mac) o la tecla <Control> (en Windows) y haciendo click con el botón izquierdo del ratón en una sección vacía de la ventana del *patch*.

[7] En este caso, es en realidad una onda coseno, como lo veremos en el próximo capítulo.

[8] Todos estos conceptos fueron explicados en la sección de teoría 1.2.

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

Luego abre el archivo **01_01.maxpat**, que encontrarás en la carpeta "Material_ Capitulos_Max_Vol 1/ Patches Max Vol 1/Patches Capitulo 01" (fig. 1.9).

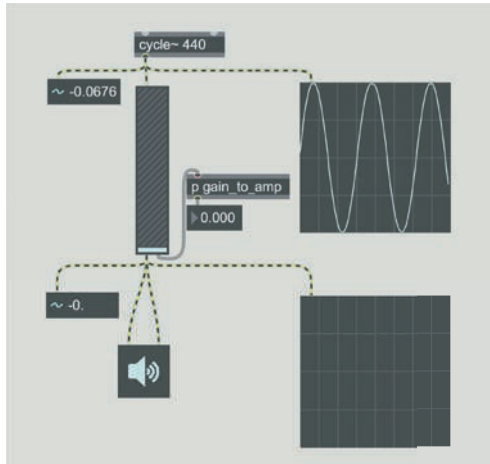


fig.1.9: archivo **01_01.maxpat**

En este archivo hemos añadido nuevos objetos al *patch* original. Los nuevos objetos a la izquierda que contienen valores numéricos son llamados objetos **`number~`**, y representan, a través del valor que muestran, una instantánea de la señal que están recibiendo. Los objetos cuadrados a la derecha se llaman objetos **`scope~`**[9], y actúan como osciloscopios en los que es posible observar una señal entrante como una forma de onda gráfica. El objeto [`p` gain_to_amp] y el objeto que está conectado a su salida (llamado **`flonum`** o *number box*) son usados para ver exactamente cuánto `gain~` está siendo aplicado para amplificar o atenuar la señal.
Una vez más, inicializa el *patch* haciendo click en el objeto `ezdac~`, y observa los cambios en los números de los objetos `number~` que se encuentran arriba a la izquierda. Estos números son producidos por el objeto `cycle~` y, si los observamos con atención, veremos que sus valores, tanto positivos como negativos, están comprendidos en el rango entre -1 y 1. En el lado superior derecho podemos observar el objeto `scope~` que nos muestra estos mismos números de forma gráfica; la mitad superior de este recuadro corresponde a números positivos, y la mitad inferior a números negativos. En el recuadro de `scope~` se muestran cientos de elementos distintos visualizados como una secuencia de puntos, en lugar del único número mostrado por el objeto `number~`. Estos puntos aparecen muy cerca unos de otros en la pantalla, lo que hace que, en conjunto, parezca una línea curva. Estos elementos, o números en la terminología de la música digital, son llamados *muestras*. La línea que oscila

[9] Los objetos `number~` y `scope~` pueden ser encontrados en la barra superior *Toolbar*, en la categoría "Audio" previamente descrita. Recuerda que se puede acceder haciendo click en el penúltimo ícono y seleccionando "Audio". Si no logras hacerlo, recuerda que puedes usar el truco que te hemos enseñado anteriormente: toma una *object box* y escribe el nombre del objeto deseado en su interior.

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

sinuosamente de arriba a abajo a través del panel del osciloscopio, es precisamente la onda sinusoidal producida por el objeto `cycle~`.

El patch también contiene un segundo objeto `number~` y un segundo objeto `scope~`, cada uno conectado al objeto `gain~`, que nos muestran, respectivamente, el número cero y una línea plana (que corresponde a una secuencia de ceros), ya que el fader de volumen está en su configuración más baja, o sea que el volumen es igual a 0.
Si movemos el fader de `gain~` hacia arriba, observamos que `number~` empieza a mostrar valores que empiezan siendo muy pequeños y que aumentan gradualmente a medida que el volumen aumenta. Simultáneamente, la línea plana del `scope~` inferior empieza su ondulación y va tomando la misma apariencia del `scope~` superior. Podemos inferir de esto que `gain~` está controlando la amplitud de la señal: mientras más subimos el fader, mayor es la amplitud de la oscilación.
Si aumentamos mucho más el valor del fader `gain~`, vemos que `number~` empieza a sobrepasar los límites de amplitud de 1 y -1, y que la forma de onda representada en el osciloscopio empieza a clipear, produciendo también un cambio en el sonido, que empieza a distorsionar.

A partir de lo que hemos visto, podemos llegar a algunas conclusiones:

1. El objeto `cycle~` produce una secuencia de valores digitales que siguen el patrón de una onda (co)seno.
2. Los límites numéricos de las muestras en la onda sinusoidal son 1 y -1. La secuencia real que estos valores siguen puede ser vista en el `scope~` superior, que muestra la forma de onda en su máxima amplitud, la cual, si es sobrepasada, distorsionará la calidad del sonido.
3. El objeto `gain~` modifica la amplitud de la onda sinusoidal, causando que los valores de las muestras en su salida sean diferentes a los recibidos en la entrada. ¿Cómo hace esto? Multiplicando los valores que recibe por una cantidad que depende de la posición del fader. Cuando este está en su posición mínima, la señal es multiplicada por 0, y el resultado es un torrente de ceros, ya que cualquier número multiplicado por 0 da como resultado 0. A medida que subimos el fader, el factor de multiplicación aumenta. Si por ejemplo, lo movemos a 0.5, las amplitudes de las muestras que ingresan al objeto `gain~` serán reducidas a la mitad (porque multiplicar un número por 0.5 es lo mismo que dividirlo por 2)[10]. Si lo llevamos

[10] Para ajustar el fader a una posición que corresponda a un factor de multiplicación de 0.5, observa el *number box* que está conectado al objeto [`p` gain_to_amp], el cual está configurado de forma precisa con el propósito de mostrar el valor del factor de multiplicación. El fader, en realidad, está dividido en incrementos logarítmicos, y usando una fórmula que explicaremos más adelante, el objeto [`p` gain_to_amp] convierte estas posiciones del fader (que pueden ser leídas desde una de las salidas del objeto `gain~`) en un valor de amplitud. Volveremos a este tema en el Interludio A, a continuación de este capítulo. Una cosa que hay que notar, es que cuando el factor de multiplicación es cercano a 0.5, la onda sinusoidal ocupa aproximadamente la mitad de la pantalla del osciloscopio.

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

a 1 (que está aproximadamente a 3/4 de la altura del fader), los valores de muestra que entran al objeto son idénticos a los que salen. Finalmente, si subimos el fader al máximo, los valores de muestra de los extremos excederán los límites de 1 y -1, aunque estas muestras volverán a los límites durante la conversión de digital a análogo. Cuando eso ocurre, la forma de onda deja de ser una onda sinusoidal, ya que resulta clipeada (como podemos ver en el osciloscopio inferior). Los valores de muestra que caen fuera del rango -1 y 1, simplemente son reasignados al valor máximo de amplitud durante la conversión, y el sonido distorsionado que escuchamos es un reflejo de la forma de onda truncada.

Hemos continuado la exploración de los conceptos de frecuencia, amplitud y forma de onda, que empezamos a ver en la sección 1.2 del capítulo de teoría. Repasemos algunos aspectos prácticos de estos conceptos básicos:

- la *amplitud* es el parámetro físico del que depende la *intensidad* de un sonido, es decir que controla si la percepción de un evento sonoro es *forte* o *piano*. En Max, el valor absoluto de la amplitud (esto es, el valor del parámetro, independiente de su signo) siempre está entre 0 y un máximo de 1;
- la *frecuencia* es el parámetro físico del que depende la altura del sonido; es el parámetro que controla si la percepción de un evento sonoro es grave o agudo. Los valores son expresados en Hertz (Hz), por lo tanto, hay que tener en cuenta que los sonidos audibles para humanos están en el rango entre 20 y cerca de 20000 Hz;
- la *forma de onda* es un parámetro fundamental para la definición del *timbre* de un sonido; este es definido como aquella característica del sonido que nos permite percibir la diferencia, por ejemplo, entre el Do central en una guitarra y la misma nota tocada en un saxofón. Recordemos que la forma de onda producida por `cycle~` es una sinusoide.

FAQ (Preguntas Frecuentes)

En esta sección trataremos de responder a algunos de los problemas más comunes a los que se enfrentan los nuevos usuarios de Max. Léelo atentamente, incluso si no has tenido ningún problema, puesto que contiene información que usarás en las siguientes secciones del libro.

1) Pregunta: He creado un objeto llamado "cycle~440", tal como se indicó en esta sección, pero el objeto no tiene entradas ni salidas. ¿Por qué?

Respuesta: asegúrate de escribir un espacio entre "cycle~" y "440", ya que el primero es el nombre del objeto, mientras que el segundo es el argumento, que en este caso, representa la frecuencia del sonido. Si las dos palabras se escriben juntas, Max buscará un objeto que no existe llamado "cycle~440", y al no encontrarlo, Max no mostrará una *object box* correcta con entradas y salidas.

2) Pregunta: De acuerdo. ¿Pero por qué Max no me dio un mensaje de error?

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

Respuesta: hay un mensaje de error que se encuentra en la **Max Console**, una ventana que el programa usa para comunicarse con el usuario. Para llamarla desde el teclado, usa <Mac: Command-m> <Win: Control-m>, o también puedes hacer click en el cuarto ícono de la barra de herramientas derecha[11]: aparecerá una ventana a la izquierda de la *Patcher Window,* en la que, probablemente, encontrarás este mensaje: "cycle~440: No such object". Si haces doble click en el mensaje de error, el objeto que lo ha causado (en este caso, el objeto "cycle~440, que no existe") será destacado en la *Patcher Window*.

3) Pregunta: He puesto un espacio entre "cycle~" y "440", pero el objeto sigue igual, sin entradas ni salidas.

Respuesta: existe un error más sutil que a menudo cometen usuarios nuevos cuando usan objetos que tienen un tilde (~) al final de su nombre. Si tienes un teclado que no posee una tecla de tilde, entonces necesitas presionar una combinación de teclas para producir el carácter (por ejemplo, <Alt-5>, en algunos Mac); puede que hayas continuado presionando alguna de estas teclas modificadoras cuando escribiste el espacio (por ejemplo, <Alt-Espacio> en Mac). La combinación resultante no es reconocida por Max, y Max no será capaz de separar el nombre del objeto de su argumento. Borra el espacio y vuelve a insertarlo, evitando presionar teclas modificadoras al mismo tiempo.

4) Pregunta: No escucho ningún sonido.

Respuesta: ¿Has hecho click en el objeto **`ezdac~`** (el ícono del pequeño parlante)? ¿Has subido el fader de volumen? ¿Estás seguro de que el sonido de la computadora no está desactivado, o logras producir sonido en algún otro programa aparte de Max? ¿Has revisado la ventana *Audio Status* (que puedes encontrar en el menú Options) para comprobar que la tarjeta de sonido correcta esté seleccionada?

UN "MANUAL DE SUPERVIVENCIA" COMPACTO PARA MAX

En esta sección te daremos un poco de información esencial para navegar fácilmente a través del ambiente Max.

COMANDOS BÁSICOS DE TECLADO

Mac <Command-n> o Windows <Control-n> creará una nueva *Patcher Window*, que es el espacio de trabajo donde implementamos nuestros *patches*.

Mac <Command-e> o Windows <Control-e> alternará entre modo edición y modo ejecución en la *Patcher Window*. En el modo de edición construiremos

[11] Cuando en la Max Console hay un error pero la ventana está escondida, el ícono de la barra de herramientas muestra una bolita roja.

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

patches a partir de objetos de la barra de herramientas superior; en el modo ejecución podemos activar *patches* e interactuar con los objetos con interfaz gráfica de usuario (como las *float number box* o el objeto `gain~`).

Mac <Command-m> o Windows <Control-m> hará aparecer la *Max Console* (si es que aún no está visible), que es una ventana usada en el ambiente Max para comunicarse con el usuario, usando breves mensajes de texto (tal como veremos más adelante).

Es posible crear algunos objetos comunes simplemente escribiendo comandos de un carácter, sin tener que usar las teclas <Command> o <Control>. El carácter 'n', por ejemplo, creará (cuando tu *Patcher* esté en modo edición) una *object box* vacía en la posición del puntero del ratón, que es exactamente lo que obtendríamos seleccionando el primer ícono de la barra superior. Existen otras teclas que también permiten la creación de objetos: trata de escribir las letras 'f', 'i', 't' y 'b' en una *Patcher Window* mientras mueves el puntero del ratón, y observa los diferentes objetos (que probablemente no signifiquen nada para ti por el momento) que son creados. Usaremos estos objetos frecuentemente en el transcurso de los siguientes capítulos.

SELECCIÓN, ELIMINACIÓN, Y COPIA
Para borrar un cable o un objeto, hay que cerciorarse de estar en el modo de edición[12], selecciona el cable o el objeto con el ratón, y luego presiona la tecla <Delete> (llamada también <*Backspace*> o <Retroceso>). Podemos seleccionar múltiples objetos a la vez haciendo click en un lugar vacío de la ventana de la *Patcher Window*, y luego arrastrando el área que se forma para que "toque" los objetos a ser seleccionados (ver fig 1.10).

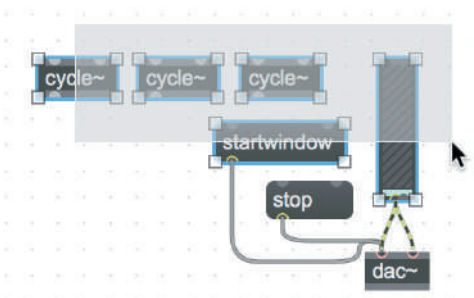


fig.1.10: selección de objetos

Si movemos uno de los objetos seleccionados, todos se moverán; de la misma forma, si presionamos la tecla <Delete>, todos los objetos seleccionados desaparecerán. Usando esta técnica de selección, los objetos serán seleccionados, pero no los cables. Si necesitamos seleccionar los cables al mismo tiempo que

12 Para asegurarte, revisa en la esquina inferior izquierda que el candado del *patch* esté abierto.

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

estamos seleccionando los objetos (para borrarlos, por ejemplo), necesitamos mantener presionada la tecla <Alt> mientras arrastramos el rectángulo de selección, y asegurarnos de que "toque" los cables que nos interesan (fig. 1.11).

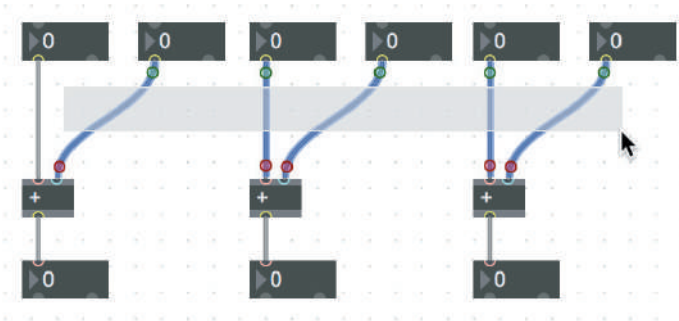


fig.1.11: selección de cables

Con la tecla <Alt> presionada, podemos también copiar un objeto. Simplemente haz click sobre este y arrástralo para crear una copia.
Si primero seleccionas múltiples objetos y luego los arrastras usando <Alt-Click>, puedes copiarlos todos (fig. 1.12).

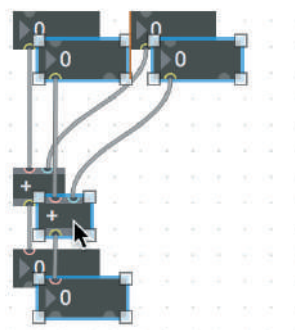


fig.1.12: copiando múltiples objetos a la vez

Si cometes un error (por ejemplo, borrar un objeto en vez de otro), puedes deshacer la acción seleccionando el comando "*undo*" desde el menú *Edit*. La repetida selección de "undo" puede cancelar una secuencia completa de comandos y regresar el *patch* al estado anterior a estas acciones. Si después te das cuenta de que no era un error (por ejemplo, que sí querías borrar el objeto), puedes restaurar las acciones usando el comando "*redo*", que también se encuentra en el menú *Edit*. Los comandos equivalentes de deshacer (*undo*) y rehacer (*redo*) son <Command-z> y <Shift-Command-z> en Mac, y <Control-z> y <Shift-Control-z> en Windows.[13]

[13] <Shift> es la tecla modificadora usada para hacer teclas mayúsculas.

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

DOCUMENTACIÓN Y AYUDA

Este es un libro autocontenido: en él encontrarás todo lo que necesitas para entender y usar los *patches* con los que ilustramos las técnicas de síntesis y procesamiento de señales en Max. El ambiente Max también provee materiales ilustrativos extensos (en inglés) a los que se puede acceder seleccionando el ítem *Reference* del menú **Help**. Cuando seleccionas este ítem, obtendrás la ventana que se muestra en la figura 1.13 (que podría tener una apariencia distinta según la versión de Max que se esté utilizando).

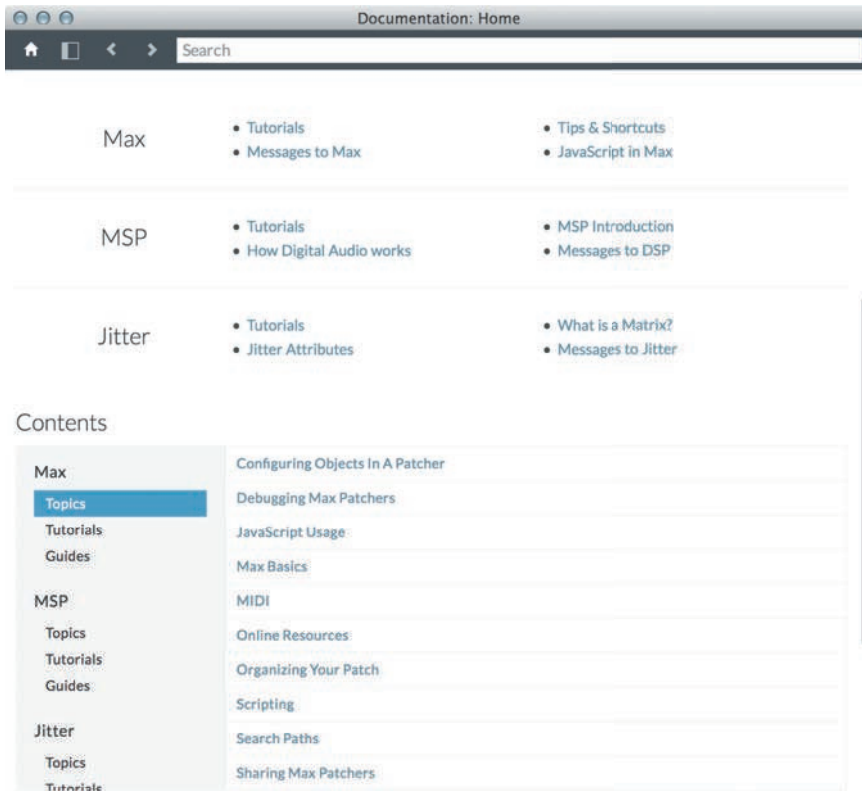


fig.1.13: ventana *Documentation*

En la figura vemos una parte de la ventana *Documentation* de Max. En la parte superior están las tres categorías principales: **Max** (el software que maneja las funciones de control - básicamente todo lo que no tenga que ver con generar o procesar audio, gráficas y/o video), **MSP** (que maneja la generación y procesamiento de señales de audio), y **Jitter** (que se encarga del procesamiento de video y gráficas, además de manejar las matrices). Para cada una de estas categorías existen Tutorials que introducen los elementos de una forma sistemática, más algunos que cubren temas centrales.

En la parte inferior hay una tabla de contenidos que contiene todos los temas (topics) por cada categoría, una lista de "Tutorials", y una serie de guías (Guides) que examinan en profundidad numerosos aspectos sobre cómo programar y operar en el ambiente Max.

© Contemponet 2020

Te aconsejamos revisar esta documentación para entender mejor su funcionamiento.

Existen también *patches* de ayuda para todos los objetos individuales de Max (también en inglés). Cuando estés en modo edición, haz <Alt-Click> en un objeto (sin arrastrarlo), y aparecerá un *patch* de ayuda relativa al objeto. Este patch será completamente funcional, y resumirá las características principales del objeto seleccionado.

Si haces <Alt-Click> sobre el objeto `cycle~` mientras estás en modo edición, por ejemplo, se abrirá el **help patch** (patch de ayuda) mostrado en la figura 1.14 (este *patch* también puede tener una apariencia distinta en distintas versiones de Max).

fig.1.14: un *patch* de ayuda

Todos los *patches* de ayuda comparten una estructura muy específica: están divididos en paneles que pueden ser mostrados haciendo click en las pestañas visibles en la parte superior izquierda de la ventana.
Cada panel individual explica algunas características o comportamientos específicos del objeto en cuestión. El número de pestañas y los nombres en ellas variarán entre objetos, con excepción de la primera y la última, que son compartidas por todos los objetos y tienen la misma etiqueta. La primera de estas pestañas compartidas lleva el nombre "basic" (básico), y destaca las características fundamentales del objeto, mientras que la última ventana compartida,

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

etiquetada con un signo de interrogación, contiene un menú cuyo primer ítem, "Open Reference", puede ser usado para abrir una página detallada del manual de referencia.
Los siguientes ítems llevan a *patches* de ayuda de objetos relacionados, que a menudo son similares al objeto en cuestión. Los últimos ítems llevan a tutorials que ilustran el uso del objeto. Si tienes un buen nivel de inglés técnico, te será útil para consultar la ayuda y para descubrir o recordar todos los detalles que necesitas, pero aunque no sepas nada de inglés, de todas maneras te recomendamos que le eches un vistazo. Ante todo, porque son *patches* funcionales que te pueden enseñar mucho sobre las técnicas prácticas y el vocabulario especializado usado en Max, el cual entre otras cosas, contiene términos como "oscillator", "frequency" o "intensity", que no son difíciles de interpretar[14].

Otra fuente útil de información es la **Clue Window** (ventana de pista), que puede ser abierta desde el menú *Window*. Esta ventana, en su configuración por defecto[15], aparece como una ventana amarilla pequeña que flota sobre las otras ventanas y muestra información relacionada con lo que se encuentra bajo el puntero del ratón. Intenta activarla y luego mueve el puntero del ratón sobre los distintos elementos de un *patch* o sobre cualquier otro de los diferentes ítems del menú. La *Clue Window* te mostrará una corta descripción de cada uno de estos ítems.

Este sistema de ayuda exhaustiva es, sin duda, uno de los puntos más fuertes del ambiente Max. Además de las ayudas (*help*) y de la *Clue Window*, recordemos el texto de "*Assistance Bubble*", que vemos cada vez que trabajamos con cables. La *Assistance Bubble* nos proporciona informaciones sobre los mensajes que pueden ser enviados o recibidos desde las entradas y salidas de un objeto. Recuerda que para verlo, necesitamos posarnos con el ratón sobre la entrada o salida de un objeto en modalidad edición (ver figs. 1.6a y 1.6b).
Otro recurso similar es el **Quickref menu** (menú de referencia rápida): abre nuevamente el archivo 01_01.maxpat, y pon el *patch* en modo edición haciendo click en el candado de la sección inferior izquierda de la *Patcher Window*. Ahora posa el puntero del ratón sobre la entrada izquierda de `cycle~` para que aparezca el círculo rojo y la *Assistance Buble*. Haciendo click derecho sobre el círculo rojo, y luego manteniendo presionado el botón del ratón, aparecerá el menú *Quickref* (ver fig. 1.15)

[14] Recuerda que en los capítulos de teoría se han incluido las traducciones del español al inglés de muchos términos técnicos.

[15] La locución **por defecto** hace referencia a un ajuste predeterminado o preestablecido de algún parámetro, función, etc., de un programa, que puede ser modificado fácilmente por los usuarios.

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

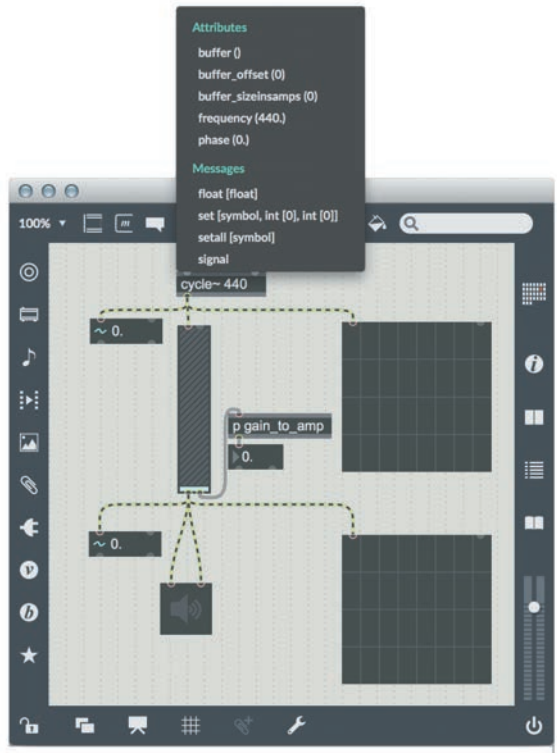


fig.1.15: Menú *Quickref*

Este menú está categorizado en dos tipos de elementos: **Attributes** (atributos), de los cuales hablaremos más adelante, y **Messages** (mensajes), que corresponden a los mensajes que un objeto dado es capaz de "entender" y usar. Seleccionando un mensaje de la lista es posible crear un nuevo objeto que se conecta "automáticamente" a `cycle~`. Haz click, por ejemplo, en el ítem *float [float]*; un nuevo objeto aparecerá ya conectado a `cycle~` (ver fig. 1.16).

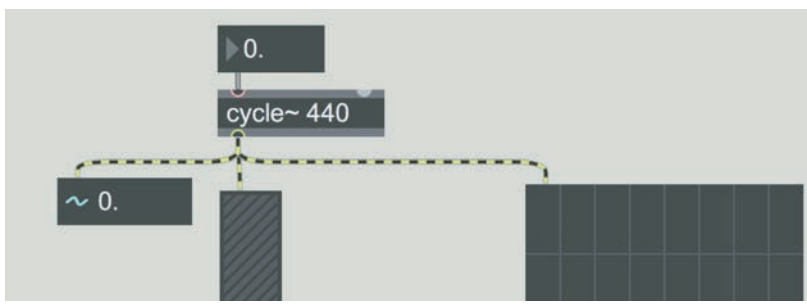


fig. 1.16: conexión de un objeto por medio de *Quickref*

Si ahora pones el *patch* en modo ejecución (cerrando con un <Click> el candado de la sección inferior izquierda), y haces pasadas de forma vertical con el ratón sobre el nuevo objeto mientras mantienes presionado el botón del

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

ratón, verás que los números cambian en la interfaz de usuario (UI) del objeto. El propósito de este objeto es almacenar e ingresar valores numéricos, y recibe el nombre de **`number`** (número), o *number box*. Los números que hemos generado deslizando el ratón sobre este objeto son enviados a **`cycle~`** para modificar su frecuencia (notarás que el argumento 440 que se encuentra en el interior de **`cycle~`** no cambia, sino que es reemplazado por los valores recién transmitidos). Ejecuta este *patch* haciendo click en el ícono de parlante y subiendo el fader a cerca de los 3/4 de su rango, luego define el número en la *number box* a valores entre 500 y 1000 usando el ratón. Escucharás que el oscilador sinusoidal produce diferentes sonidos correspondientes a las frecuencias que se muestran en la *number box*. Trataremos con más detalle la *number box* en las siguientes secciones de este capítulo.

Si ahora mueves el puntero del ratón al lado izquierdo de un objeto (en modo de edición, por supuesto) verás que aparece un círculo amarillo que contiene un triángulo pequeño (fig. 1.16b). Si haces click en este ícono, aparecerá un nuevo menú contextual, llamado **Object Action Menu** (menú de acción de objeto).

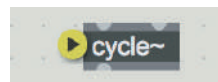


fig. 1.16b: icono *Object Action*

Este menú (fig. 1.16c) contiene los elementos más útiles para administrar, transformar y encontrar información sobre un objeto. En la práctica, todos los ítems de este menú también pueden encontrarse de otros modos (los estudiaremos poco a poco, a medida que los necesitemos): este menú tiene la ventaja de ubicarlos a todos en un lugar donde pueden ser invocados con un solo click.

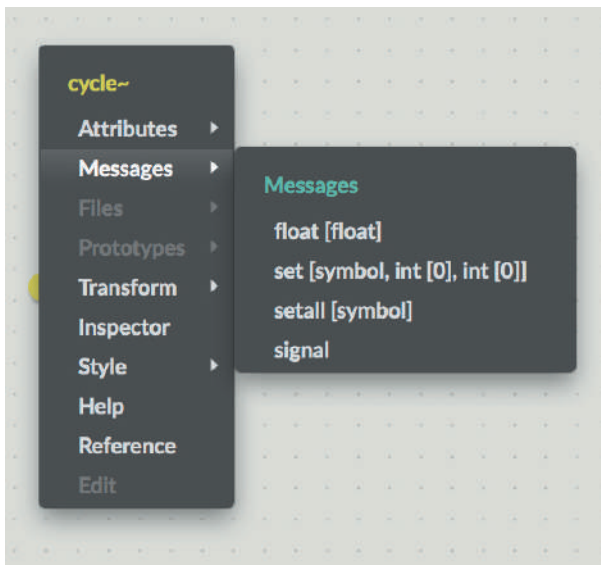


fig. 1.16c: Menú *Object Action*

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

BARRAS DE HERRAMIENTAS DE LA PATCHER WINDOW (VENTANA DE PATCHER)

Ahora examinaremos con mayor detalle las cuatro *Toolbars* (barras de herramientas), que, como hemos dicho, rodean la *Patcher Window* (que es la ventana en la que escribimos nuestros programas en Max). La **Barra de Herramientas Superior** contiene los objetos de interfaz disponibles en Max, agrupados por categorías (ver fig. 1.17), además de un menú de Zoom en la esquina superior izquierda, con el que es posible aumentar o reducir la visualización del *patch*.

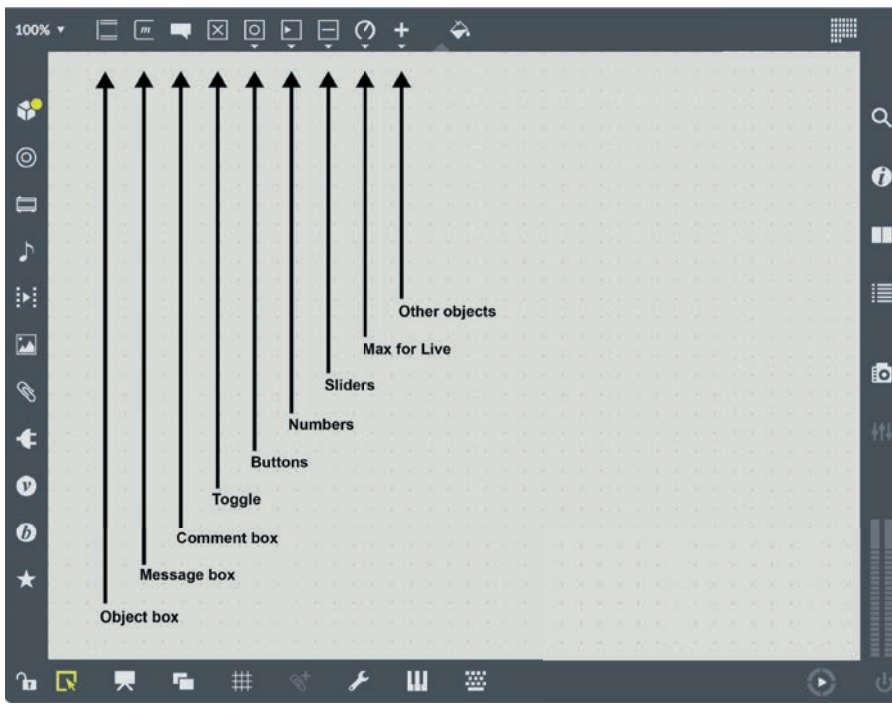


fig.1.17: Barra de Herramientas superior

Los primeros cuatro íconos después del menú Zoom corresponden a cuatro objetos de uso frecuente: la *object box* (con la que ya estás familiarizado), la *message box*, la *comment box*, y el objeto `toggle`, que introduciremos dentro de poco. Le siguen cinco íconos, de los cuales es posible abrir el mismo número de paletas; cada una de estas contiene objetos de una categoría predeterminada:

Buttons: esta paleta contiene objetos de interfaz que funcionan como botones.
Numbers: esta paleta contiene objetos de interfaz que muestran y generan valores (como la *number box* mostrada en la fig. 1.16).
Sliders: esta paleta contiene objetos de interfaz que funcionan como cursores; en otras palabras, producen valores dentro de un rango cuando los arrastras usando el ratón (ya vimos el uso de esta paleta en la fig. 1.5).
Max for Live: esta paleta contiene los objetos de interfaz para Max for Live (que trataremos en el segundo volumen).

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

Add (otros objetos): esta paleta contiene todos los otros objetos relevantes, y está subdividida en las subcategorías: *Basic* (básico), *Audio*, *Data* (datos), *Images* (imágenes), *Interface* (interfaz) y *Jitter*.
Ya hemos visto la subcategoría de *Audio* en la fig. 1.7; exploraremos los contenidos de otras categorías a medida que las necesitemos.

Hay un ícono al final de la barra de herramientas superior que representa un tarro de pintura. Este ícono no sirve para crear objetos, sino para configurar la apariencia gráfica de la *Patcher Window* y de los objetos de Max contenidos en la misma: un solo click en este ícono y aparecerá la *Format Palette* (paleta de formato). No nos ocuparemos de los detalles de apariencia durante esta primera sección; de todas maneras, intenta explorar un poco este ícono: selecciona por ejemplo, en modo edición, una *object box* dentro del *patch* 01_01.maxpat y usando la *Format Palette*, trata de modificar su color, o el tamaño y tipo de carácter.

En la parte derecha de la Barra de Herramientas superior hay un último ícono que abre un calendario; a través de este es posible activar en orden cronológico las patches en las que hemos trabajado.

Usando la **Barra de Herramientas Izquierda** es posible acceder a cualquiera o a todos los tipos de archivos usados en la programación de Max (fig. 1.17b).

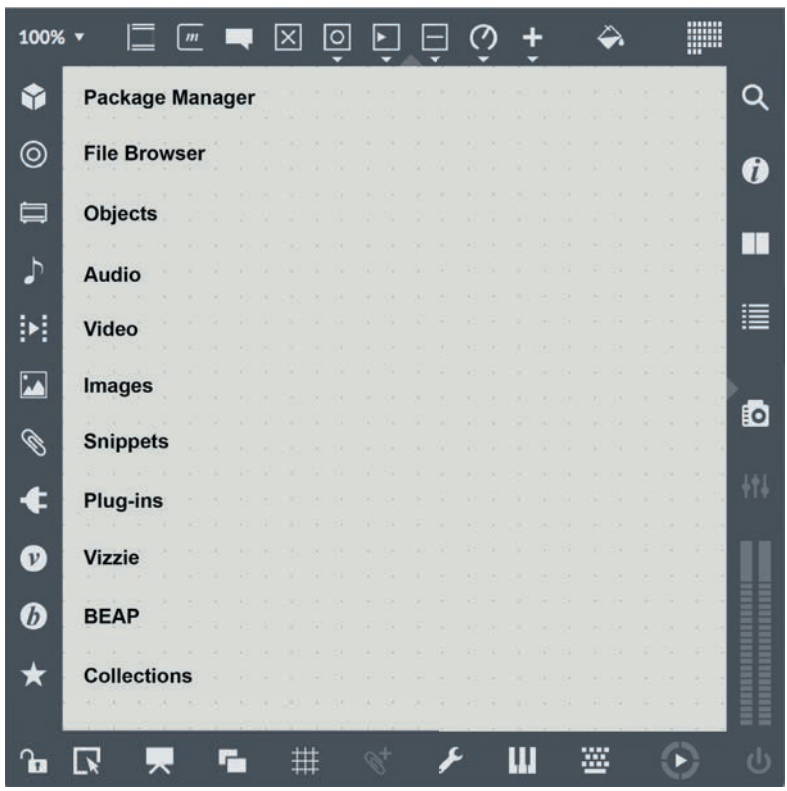


fig.1.17b: Barra de Herramientas izquierda

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

Por ahora vamos a obviar los primeros dos íconos que introducen, respectivamente, el *Package Manager* y el *File Browser* (navegador de archivos). Haz click en el tercero: aparecerá un menú lateral que contiene todos los objetos disponibles (no solo los objetos interfaz, disponibles en la barra de herramientas superior). Los objetos de este menú están divididos en categorías y subcategorías, y al hacer doble click sobre el nombre de un objeto, este aparecerá en el *patch*. De forma alternativa, puedes arrastrar el nombre a la *Patcher Window*, haciendo que aparezca la instancia de un objeto (estas dos formas alternativas de crear objetos están disponibles en todos los menús de las barras de herramientas).

El cuarto ícono nos permite ubicar y usar archivos de audio dentro de un *patch*; el quinto, archivos de video, y el sexto, imágenes. Ten en cuenta que, para poder estar visibles en estos menús, estos archivos deben encontrarse en el *File Search Path* (ruta de búsqueda de archivos) de Max, que revisitaremos más adelante. Ahora, con el ícono de candado abierto, trata de arrastrar algunos archivos del menú de audio al área de programación, y observa lo que ocurre; revisitaremos este menú en la sección 1.5.
El séptimo ícono (representado como un *clip*) nos permite acceder a *Snippets*, que no son otra cosa que fragmentos reusables de código que pueden ser memorizados usando la Barra de Herramientas inferior (ver más abajo).
El octavo ícono nos da acceso a todos los plug-ins de Max for Live, VSTs, y Audio Units que hemos instalado en nuestra computadora. Por el momento, ignoraremos los íconos restantes.

La **Barra de Herramientas Inferior** contiene herramientas usadas durante la programación de Max. El primer ícono es el candado, que como ya sabes, nos permite pasar del modo edición al modo ejecución, y vice-versa. Discutiremos cada uno de los otros íconos de esta barra de herramientas en el momento apropiado; por el momento ocupémonos del sexto: un clip con un signo positivo, similar al que se encuentra en la barra de herramientas izquierda. Este ícono nos permite crear nuevos *Snippets*. Para entender cómo funcionan, abre el *patch* llamado 01_01.maxpat (fig 1.9), pon la *Patcher Window* en modo edición, y selecciona los objetos `gain~` (los faders verticales), el objeto `ezdac~` (el parlante pequeño), y el objeto [`p` gain_to_amp].[16] En este punto, haz click en el ícono de clip de la barra de herramientas inferior: aparecerá una pequeña ventana en la que puedes darle el nombre al *snippet* creado: escribe "audiostart" y presiona <Enter>. A continuación, crea un nuevo *patch* y haz click en el sexto ícono de la barra de herramientas izquierda (el otro clip). Aparecerá un menú que, entre sus entradas, ahora contiene un ítem llamado "audiostart". Arrastra este elemento a la *Patcher Window* y verás aparecer el snippet que previamente grabaste. A través de esta técnica es posible capturar secciones de código de uso frecuente para ser reusadas. Como puedes ver, este menú ya contiene algunos *snippets* predefinidos, listos para ser usados; puedes crear otros fácilmente mientras trabajas con Max.

[16] Para seleccionar múltiples objetos puedes usar la técnica mostrada en la fig 1.10, o hacer click en los objetos, uno tras otro, mientras mantienes presionada la tecla shift.

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

Pasemos ahora a la **Barra de Herramientas Derecha**. Todos los otros íconos activan la **Sidebar**, una especie de barra lateral, una ventana que se abre en la parte derecha de la *Patcher Window* y que contiene una variedad de herramientas e información útiles. Abre, por ejemplo, el *patch* 01_01.maxpat (fig. 1.9), ponlo en modo edición, selecciona el objeto `ezdac~` (el pequeño parlante), y haz click en el segundo ícono (mostrado como la letra "i" inscrita en el círculo). Al hacer esto se abrirá la *Sidebar* (barra lateral) y mostrará el *inspector* del objeto seleccionado. El inspector contiene todos los atributos que definen la apariencia y el comportamiento de un objeto. Hablaremos de esto con mayor detalle más adelante. El tercer ícono permite el acceso rápido a la documentación sobre el objeto seleccionado, mientras que el cuarto abre la *Max Console*, que como sabemos, es la ventana a través de la cual Max muestra mensajes de error e información útil que veremos más adelante.
Hablaremos de los otros íconos en su momento.

Al final de esta barra de herramientas se pueden ver los controles combinados de medición y ganancia de Audio, con los cuales podemos manejar el volumen de audio global del *patch*, además de un botón de encendido/apagado de audio (*Audio On/Off*), que puede ser usado para activar y desactivar la producción de señales de audio en el *patch* (de la misma forma que puede hacerlo un objeto `ezdac~`).

UN POCO DE ORDEN
Probablemente has notado que algunos de los cables presentes en el *patch* 01_01.maxpat (fig. 1.9) presentan ángulos que dividen sus conexiones en segmentos ordenados. ¿Sabes cómo se hace esto? La respuesta es simple: seleccionando **Segmented Patch Cords** (cables segmentados) en el menú *Options*. Si usas esta opción, el procedimiento para conectar dos objetos será ligeramente distinto a lo que hemos visto hasta ahora: haz click en la salida que será la fuente de la conexión y luego arrastra el cable hacia afuera del objeto sin mantener presionado el botón del ratón. El cable seguirá conectado al puntero del ratón. Los segmentos pueden ser creados en cualquier momento con un click del ratón en el punto en que quieres cambiar de dirección: por cada click, se crea un nuevo segmento. El último click debe hacerse en la entrada del objeto que quieres conectar.

Si cometes un error y quieres soltar el cable sin hacer la conexión, necesitas presionar <Command-click> en Mac o <Control-click> en Windows, o presionar la tecla de escape <Esc>.

Si seleccionas algunos objetos que estén alineados aproximadamente de forma horizontal, y luego presionas <Command-y> en Mac, o <Control-Shift-a> en Windows, los objetos se alinearán perfectamente. El mismo comando funciona para crear columnas verticales de objetos, uno sobre otro (los dos objetos `scope~` y los dos objetos `number~`, en 01_01.maxpat, fueron alineados de esta forma).
Los objetos pueden ser alineados fácilmente usando la función **Snap to Object** (alinear al objeto), que está activada por defecto. Cada vez que posicionas un

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

objeto en un *patch*, esta función alineará el objeto con los otros objetos circundantes. Otra función útil es **Distribute** (distribuir), que se encuentra en el menú *Arrange* (organizar); esta función hace que sea posible distribuir un grupo de objetos seleccionados en intervalos iguales de forma horizontal o vertical.

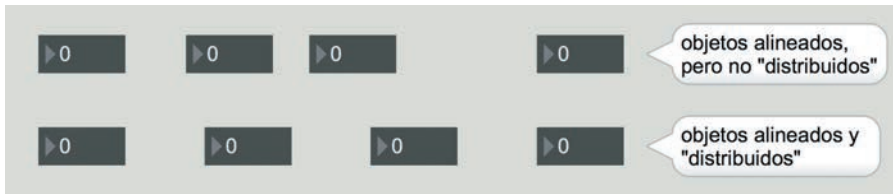


fig. 1.18: función *Distribute*

A veces puede ocurrir que un *patch* complejo esté muy poblado, con docenas de objetos y cables entre ellos. En este caso, es posible hacer que un conjunto de objetos y cables estén invisibles durante el modo ejecución (pero no en modo edición, por obvios motivos). Para esconder uno o más objetos y cables, hay que seleccionarlos, en modo edición y luego presionar <Command-k>, en Mac, o <Control-k>, en Windows. Al pasar al modo ejecución, los objetos desaparecerán.

Para revertir esta acción y hacer que los objetos reaparezcan durante el modo ejecución, usa <Command-l/Control-l> (letra "ele" minúscula) durante el modo edición. Es posible seleccionar y esconder múltiples objetos a la vez usando estos comandos. De manera alternativa, podemos seleccionar del menú *Object* los ítems **Hide on Lock** (esconder al cerrar candado) y **Show on Lock** (mostrar al cerrar candado). Practica con el archivo 01_01.maxpat haciendo que varios objetos y cables aparezcan y desaparezcan. Existe una manera aún más eficiente de ordenar *patches*, y consiste en usar el modo presentación. Explicaremos este modo más adelante, cuando ya hayamos construido *patches* más complejos.

ACTIVIDAD

Crea una nueva *Patcher Window* y trata de reproducir el *patch* contenido en el archivo **01_01.maxpat**.
Usa el *snippet* que grabaste en el párrafo anterior (si no lo has hecho, ¡hazlo ahora!), porque de otra forma sería imposible crear el objeto [`p` gain_to_amp]. ¡Presta especial atención a no confundir el objeto `number~` con la *number box*! Si no has logrado encontrar `scope~` y `number~` en la barra de herramientas superior de la *Patcher Window*, recuerda que siempre puedes tomar una *number box*, escribir el nombre del objeto en su interior, y transformarla en el objeto gráfico que buscas. Observa que la forma de onda mostrada en el osciloscopio que has creado es diferente de la que se encuentra en el archivo original. En la siguiente sección veremos por qué.

...

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

El capítulo continúa con:

1.2 FRECUENCIA, AMPLITUD Y FORMA DE ONDA
Generadores de banda limitada

1.3 VARIACIONES DE FRECUENCIA Y AMPLITUD EN EL TIEMPO: ENVOLVENTES Y GLISANDOS
Glisandos
Envolventes
Modo presentación (presentation mode)
Curvas exponencial y logarítmica

1.4 RELACIÓN ENTRE FRECUENCIA E INTERVALO MUSICAL, Y ENTRE AMPLITUD Y NIVEL DE PRESIÓN SONORA
Glisandos naturales
Conversión de decibeles a amplitud

1.5 NOTAS SOBRE EL TRABAJO CON SONIDO MUESTREADO
Administrar archivos con el explorador de archivos
Grabar un archivo de audio
Leer un sonido desde un buffer de memoria

1.6 NOTAS SOBRE PANNING

1.7 ALGUNOS FUNDAMENTOS DE MAX
¿Cables grises o amarillos con negro? Max vs. MSP
Orden de ejecución en Max
El objeto panel y los niveles de fondo

ACTIVIDADES
• Sustitución de partes de algoritmos, Corrección, Compleción y análisis de algoritmos, Construcción de nuevos algoritmos

EVALUACIÓN
• Proyecto de ingeniería en reversa

MATERIALES DE APOYO
• Lista de comandos principales de Max - Lista de objetos de Max - Lista de mensajes, atributos y parámetros de objetos de Max específicos - Glosario

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

Interludio A
PROGRAMANDO CON MAX

IA.1 MAX Y LOS NÚMEROS
IA.2 GENERACIÓN DE NÚMEROS ALEATORIOS
IA.3 MANEJO DEL TIEMPO: EL OBJETO METRO
IA.4 SUBPATCHES Y ABSTRACCIONES
IA.5 OTROS GENERADORES DE NÚMEROS ALEATORIOS
IA.6 ORDEN DE MENSAJES CON TRIGGER
IA.7 OBJETOS PARA ADMINISTRAR LISTAS
IA.8 LA MESSAGE BOX Y ARGUMENTOS VARIABLES
IA.9 ENVÍO DE SECUENCIAS DE BANGS: EL OBJETO UZI
IA.10 SEND Y RECEIVE
IA.11 USO DE ATRIBUTOS DENTRO DEL OBJETO
IA.12 AUDIO MULTICANAL

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri

© Contemponet 2020

AGENDA DE APRENDIZAJE

PRERREQUISITOS DEL CAPÍTULO

- Contenidos del Capítulo 1 (Teoría y Práctica)

OBJETIVOS

Habilidades

- Saber usar las funcionalidades básicas de Max pertinentes a números enteros y de punto flotante
- Saber generar y controlar secuencias de números aleatorios, opcionalmente con el uso de un metrónomo
- Saber construir objetos embebidos y abstracciones
- Saber repetir mensajes a través de múltiples salidas de objetos
- Saber ensamblar y manipular listas, usando métodos tanto gráficos como no-gráficos
- Saber usar argumentos variables
- Saber administrar comunicación entre objetos sin el uso de cables

CONTENIDOS

- Números enteros y floating point en Max
- Generación y control de números aleatorios con los objetos random , drunk, etc.
- Generación de eventos rítmicos regulares usando el objeto metro
- Construcción de subpatches y abstracciones
- Administración de argumentos de lista y variables
- Uso de objetos send y receive para comunicación inalámbrica entre objetos

TIEMPO DE DEDICACIÓN - Capítulo 1 (Teoría y Práctica) - Interludio A

Autodidactas

Sobre un total de 300 horas de estudio individual (volumen I, teoría y práctica):

- Aproximadamente 100 horas

Cursos

Sobre un total de 60 horas de clase + 120 de estudio individual (volumen I, teoría y práctica):

- Aproximadamente 16 horas de clase frontal + 4 de "feedback"
- Aproximadamente 40 horas de estudio individual

ACTIVIDADES

- Sustitución de partes de algoritmos, corrección, compleción y análisis de algoritmos, construcción de nuevos algoritmos.

EVALUACIÓN

- Proyecto integrado: Ingeniería en reversa

MATERIALES DE APOYO

- Lista de objetos Max - Lista de mensajes, atributos y parámetros de objetos Max específicos - Glosario de términos usados en este capítulo

© Contemponet 2020

En este primer "Interludio" examinaremos en mayor profundidad unos pocos aspectos de programación de Max: se trata de información esencial que será útil para la lectura de este libro. Por consiguiente, te alentamos a no saltar esta sección a menos que realmente seas un experto en Max. Es importante que implementes todos los *patches* tutorial que proponemos en el texto, ya que estos pequeños esfuerzos arrojan los mayores resultados.

IA.1 MAX Y LOS NÚMEROS: LOS OPERADORES BINARIOS

Como cualquier lenguaje de programación que se respete, Max puede hacer muchas cosas con números. Empezaremos este capítulo revisando los operadores más simples, los de adición, sustracción, multiplicación y división.

SUMA DE NÚMEROS ENTEROS

Recrea el *patch* simple mostrado en la figura IA.1 (y asegúrate de dejar un espacio entre "+" y "5").

fig. IA.1: suma

El objeto + suma su argumento (que es 5, en este caso,) a cualquier número recibido en su entrada izquierda. Si alimentamos otros números al objeto a través de la *number box* conectada sobre ella (seleccionándola, por ejemplo, en modo ejecución y usando las teclas de flecha en el teclado para cambiar este valor), podemos seguir los resultados de la operación en la *number box* inferior. La entrada derecha de + cambia el argumento, y si ingresamos un número en esta entrada usando otra *number box*, el número será sustituido por el argumento del objeto + en la suma que será producida cuando nuevos números sean ingresados a través de la entrada izquierda. Comprueba esto añadiendo una *number box* a la derecha, como se muestra en la figura IA.2:

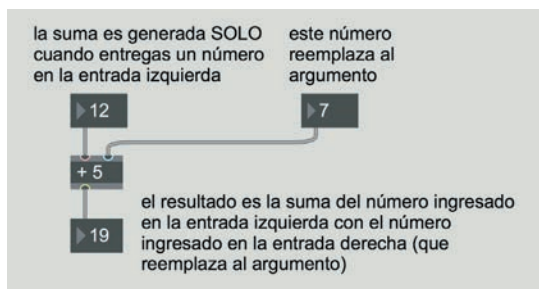


fig. IA.2: suma con un argumento variable

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

Observa que la operación + es ejecutada solo cuando se envía un número a la entrada izquierda, y nunca cuando un número es enviado a la entrada derecha. Cada vez que un nuevo número es enviado a una de las dos entradas, una variable interna correspondiente a esa entrada es actualizada. Los contenidos antiguos de la variable son borrados y reemplazados por el nuevo valor. Adicionalmente, un número enviado a la entrada izquierda gatilla la operación +, sumando los contenidos de las dos variables internas y emitiendo el resultado. Esto es un patrón común: una gran mayoría de objetos de Max ejecuta sus funciones y emite resultados solo cuando un mensaje es recibido en su entrada izquierda. Los mensajes que entran a otras entradas sirven para modificar argumentos actualizando estructuras de datos internos, o para modificar el comportamiento de los objetos sin causar resultados visibles.
En el léxico de los objetos de Max, la entrada izquierda es definida como una entrada "caliente", y hace que la operación sea efectuada en añadidura a la actualización de los valores internos correspondientes a la entrada. La mayoría de las otras entradas son definidas como entradas "frías" que actualizan variables internas sin producir salida. Observa que el círculo que aparece alrededor de la entrada de un objeto cuando posas el ratón sobre ella, en el modo de edición, es rojo para las **entradas** "**calientes**" y azul para las **entradas** "**frías**".
¿Pero existe una manera de actualizar el valor de la suma cuando enviamos un número a la entrada derecha de un objeto suma? En otras palabras, ¿existe una manera de convertir una entrada "fría" en una que se comporte como "caliente"? claro que sí; podemos examinar esta técnica en la figura IA.3:

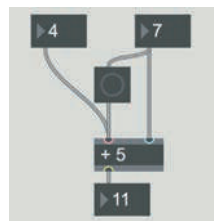


fig. IA.3: cómo hacer que una entrada "fría" se comporte como una entrada "caliente"

En este *patch* primero enviamos un número a la entrada derecha y luego un *bang* a la entrada izquierda (recuerda que el orden para los mensajes de Max es de derecha a izquierda). Dado que `button` transforma cualquier cosa que reciba en un mensaje *bang*, un número en la entrada derecha producirá este tipo de mensaje, el cual, como ya sabes, forzará al objeto que lo recibe a producir un resultado, que en este caso es nuestra suma.
¿Qué valores son sumados? Los números de las variables internas del objeto, que en este caso, son el último número enviado a la entrada derecha (en la figura, el número 7) y el último número enviado a la entrada izquierda (correspondiente a 4 en la figura)[1].

[1] Recuerda que en esta figura, tal como en las siguientes, el argumento "5" original es reemplazado por cualquier número enviado a la entrada derecha.

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

En otras palabras, con este sistema una entrada "fría" del objeto suma se comporta como si fuera una entrada "caliente". Trata de reconstruirlo, verificando que la *number box* de la derecha gatille un resultado cada vez que su valor sea modificado.
Es esencial que las posiciones de los objetos en tu *patch* sean absolutamente las mismas de la figura. Si posicionas el `button` a la derecha del objeto `+,` por ejemplo, se producirán resultados no deseados, ya que el *bang* se disparará antes de que el nuevo valor sea almacenado, causando que el objeto + use el valor antiguo que había sido copiado a la variable interna en su cálculo, en vez del valor nuevo (como se muestra en la figura IA.4).

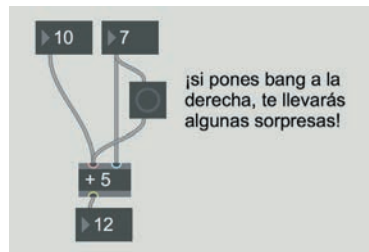


fig. IA.4: resultados erróneos producidos por un error en el orden de la ejecución

Borra las dos *number box* sobre el objeto + y conecta una *message box* a su entrada izquierda. Posteriormente, escribe dos números (separados por un espacio) en el interior de la *message box*, como se muestra en la figura IA.5:

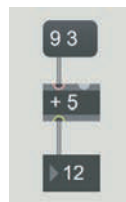


fig. IA.5: sumando una lista

Si ahora haces click en la *message box* (en modo ejecución), el objeto + sumará sus valores: el operador se comporta como si hubiera recibido el segundo número en su entrada derecha (y, como es usual, el argumento 5 de la *number box* es reemplazado por ese nuevo valor). Este comportamiento (la habilidad de aceptar listas de todos o más elementos en su entrada izquierda y luego rutear los ítems individuales de la lista a las otras entradas) es también una característica común de muchos objetos de Max, y funciona no solo con operadores binarios, sino también, a menudo, con objetos que tienen tres o más entradas.

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

Para ver una aplicación musical simple del objeto suma de Max, abre el archivo **IA_01_transposicion.maxpat** (fig. IA.6).

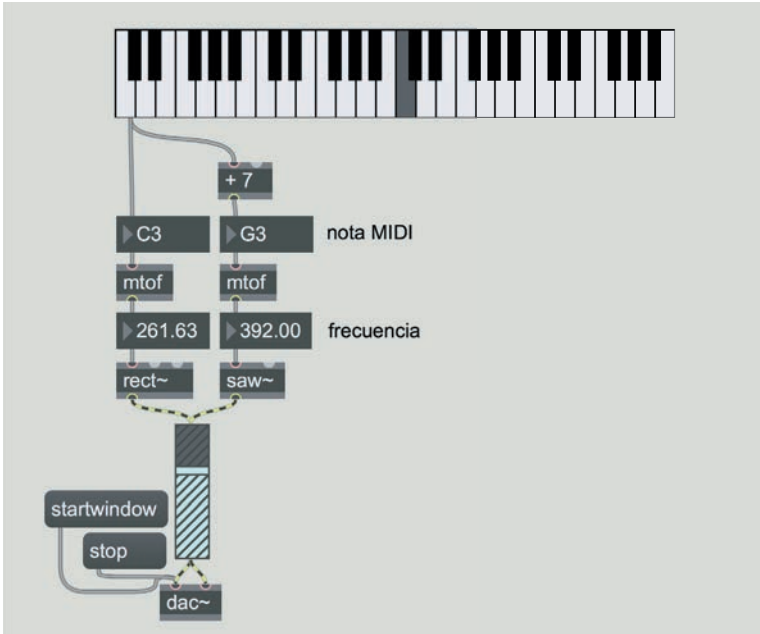


fig. IA.6: archivo **IA_01_transposicion.maxpat**

Cada vez que una tecla es presionada en el objeto `kslider` de este *patch*, dos notas son generadas, la segunda de las cuales se encuentra 7 semitonos sobre la primera (es decir, una quinta arriba). Cada vez que presionas una tecla en el `kslider`, el valor del número de nota MIDI correspondiente (un DO en la figura IA.6, por ejemplo) es enviado al objeto `mtof` de la izquierda, que lo convierte a un valor de frecuencia. Al mismo tiempo, el número de nota es enviado también a un objeto + que le suma el valor de 7 (produciendo el número de nota MIDI de un SOL, según el ejemplo de la figura IA6). La suma resultante es enviada al objeto `mtof` de la derecha del *patch*.
Usando el objeto suma de esta manera, es posible transponer notas MIDI en intervalos arbitrarios de nuestra preferencia. En este ejemplo, después de que los valores MIDI de las dos notas son convertidos en frecuencias, son enviados a dos osciladores, `rect~` y `saw~`, que suenan a distancia de un intervalo de quinta. Trata de modificar el *patch* para usar otros intervalos (una tercera, o 4 semitonos; una quinta, o 5 semitonos; una octava, o 12 semitonos; etc.). Posteriormente, prueba añadiendo otro objeto suma conectado a otro objeto `mtof` que a su vez esté conectado a un oscilador, para que cada vez que se presione `kslider` produzca un acorde mayor de tres notas. Por ejemplo, si se presiona la tecla C2 (la tecla que se encuentra una octava abajo del DO central), se obtiene el acorde de DO MI SOL de la segunda octava (C2 E2 G2).

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

NÚMEROS CON UN PUNTO DECIMAL, Y OTRAS OPERACIONES

Hasta ahora solo hemos usado números enteros. Cuando al operador + se le da un entero como argumento (o no tiene argumento), Max asumirá que deseas realizar cálculos matemáticos con números enteros. Para hacer que el objeto funcione con números de punto flotante, como se muestra en la figura IA.7, necesitamos conectar *float number box* a sus entradas; como puedes ver, el argumento es "0" (el punto decimal es importante, pero no es necesario poner ningún número a la derecha del punto decimal). El punto decimal en el argumento declara al objeto + que pretendemos usar números no-enteros en sus entradas, en otras palabras, que pretendemos usar matemática de punto flotante. Duplica el *patch* por tu cuenta y realiza algunas sumas usando números de punto flotante (recuerda que la entrada "caliente" es la izquierda).

fig. IA.7: sumando números que contienen puntos decimales

Todo lo que hemos tratado hasta este punto con relación a la suma también se aplica a la resta, la multiplicación y la división. Construye el *patch* mostrado en la figura IA.8 y verifica su funcionamiento.

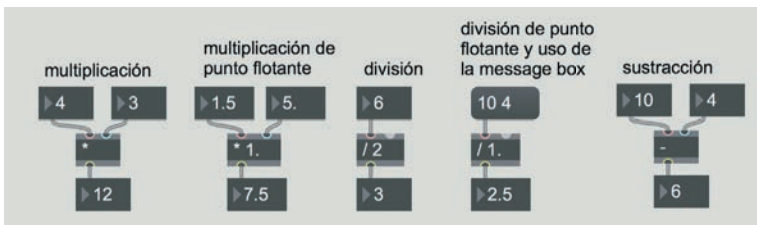


fig. IA.8: otros operadores matemáticos

Aunque estos ejercicios puedan parecerte triviales, es recomendable hacerlos, puesto que te ayudarán a darte cuenta de los detalles y particularidades del ambiente Max, detalles cuya importancia será relevada en los *patches* que usaremos en los siguientes capítulos.
Por ejemplo, en el caso del comentario "división de número de punto flotante y uso de la *message box*" (fig. IA.8), vemos que estamos evaluando una lista de números enteros ("10 4"). No obstante se entreguen números enteros, el operador ya fue declarado para representar una operación con punto flotante gracias al argumento de punto flotante provisto en su *number box*, y como consecuencia, el resultado emitido será un número de punto flotante.
Usemos de nuevo estas simples operaciones en un contexto musical. Como ya sabes, la distancia entre dos notas, es decir, un intervalo, puede ser expresada como la razón entre ambas frecuencias. Por ejemplo, el intervalo de una quinta,

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

correspondiente a 7 semitonos, puede ser expresado usando la razón 3/2. En otras palabras, dada una frecuencia (digamos 261.63 Hz, que corresponde al DO central), si multiplicamos esta frecuencia por 3 y la dividimos por 2, obtendremos la frecuencia de la nota que se encuentra una quinta arriba (en este caso sería 392.44 Hz, correspondiente a la nota SOL). Para ver esto, abre el archivo **IA_02_quintas.maxpat** (fig. IA.9).

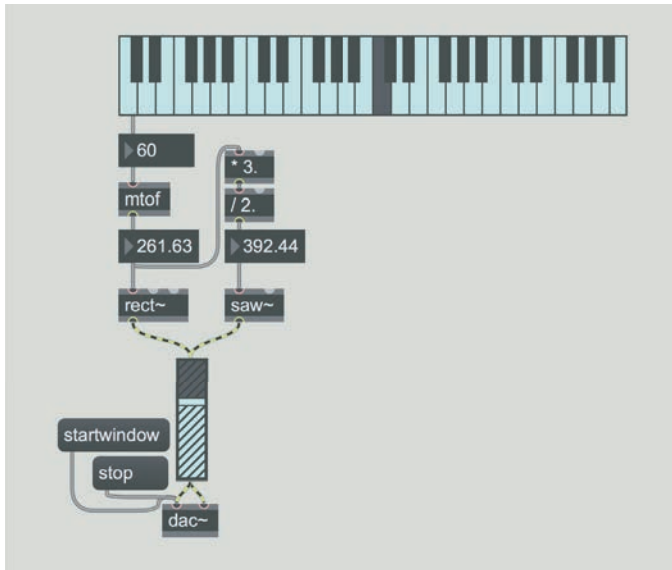


fig. IA.9: archivo **IA_02_quintas.maxpat**

En el *patch* vemos que la nota generada por el objeto `kslider`, a través de `mtof`, es transformada en una frecuencia, y es enviada al oscilador `saw~`. La salida de `mtof` también es multiplicada por 3/2, obteniendo de esta forma la frecuencia de la nota que está una quinta arriba, y luego es enviada al segundo oscilador `saw~`. Observa que los argumentos del objeto multiplicador y del objeto divisor tienen un punto decimal como último carácter, lo que declara a Max que debería usar operaciones de punto flotante. Observa también el ruteo de los dos cables que salen de la *float number box* posicionada bajo el objeto `mtof`: un cable está conectado directamente al objeto `saw~` que se encuentra justo bajo este, pero el segundo sigue a la derecha, luego sube de nuevo, y finalmente termina conectado al objeto multiplicador que tiene el argumento 3.

Al comparar las figuras IA.6 y IA.9, observamos que las frecuencias calculadas para la nota que se encuentra una quinta arriba son diferentes. Usando dos objetos `mtof` distintos en el primer *patch*, hemos calculado una quinta temperada (la quinta que normalmente se usa en la música occidental), la cual se define como el intervalo formado por 7 semitonos temperados. Al usar en cambio la razón 3/2 en la salida de un solo `mtof`, estamos calculando una quinta natural, que es un poco más ancha que una quinta temperada. La diferencia es aproximadamente de 2 cents (una unidad definida como 1/100 de un semitono temperado).

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

OPERACIONES QUE USAN EL SIGNO DE EXCLAMACIÓN

Todos los operadores de los que hemos hablado hasta ahora son operadores binarios, llamados así porque necesitan dos entradas (también llamadas operadores en la jerga computacional) para producir su salida. En los objetos que ya hemos visto, el primer operando es el número que entra por la entrada izquierda y que gatilla la operación, mientras que el segundo operando es el argumento (o el número de la entrada derecha).
Para la resta y la división, sin embargo, existen dos objetos para los que los operandos están invertidos. Su segundo operando entra por la entrada izquierda y gatilla la operación, mientras que el primer operando es el argumento del objeto. El nombre de estos objetos "invertidos" está formado por un signo de exclamación y por el signo de la respectiva operación: **`!-`** para la resta (que ya hemos visto en el *patch* 01_17_pan.maxpat), y **`!/`** para la división. Observa la figura IA.10.

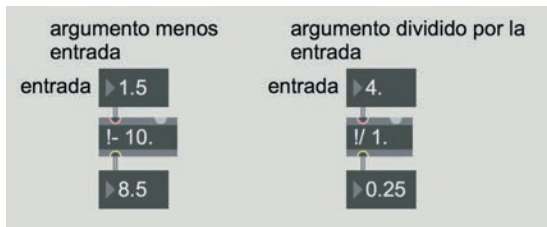


fig. IA.10: operadores que contienen un signo de exclamación

En el primer caso, el valor de entrada (1.5) es restado al argumento (10), y el resultado es 8.5 (10 - 1.5). En el segundo caso, el argumento (1) es dividido por el valor ingresado, lo que da como resultado 0.25, ya que 1/4 = 0.25. Los operandos están invertidos con respecto al orden normal de la resta y la división. Construye este *patch* y compara estas operaciones con las operaciones análogas, pero sin signo de exclamación. Ya nos hemos encontrado con el operador **`!-`** en acción, en el *patch* **01_17_pan.maxpat** de la sección 1.6.

Todos estos operadores también existen en versiones MSP que generan señales **`(+~, -~, *~, /~),`** que ya hemos usado algunas veces en el capítulo anterior. Los operadores MSP requieren que haya una señal presente en una de los dos entradas (usualmente la izquierda) y pueden recibir o una señal o valores numéricos en su otra entrada (también pueden usar un valor único provisto como argumento). Para más información sobre operadores y objetos, recuerda que si haces <Alt-Click> en modo edición, aparecerán los *patches* de ayuda.

LOS OBJETOS INT Y FLOAT

Existen dos objetos que te permiten almacenar valores y recuperar estos valores después usando un *bang*. Estos objetos son **`int`** (para el almacenamiento de números enteros) y **`float`** (para números de punto flotante). En la figura IA.11, vemos que estos objetos poseen dos entradas: si un valor es enviado a la entrada izquierda (caliente), es almacenado y transmitido inmediatamente, mientras que si un valor es enviado a la entrada derecha (fría), es almacenado

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

pero no transmitido. Para recuperar el valor grabado en cualquier momento de cualquiera de estos objetos, se puede enviar un *bang* a la entrada izquierda, lo que causará que el valor sea transmitido. En ambos casos, los valores de entrada son copiados a la memoria del objeto (y por lo tanto, pueden ser recuperados con un *bang* cuando lo deseemos) hasta que nuevos valores tomen su lugar.

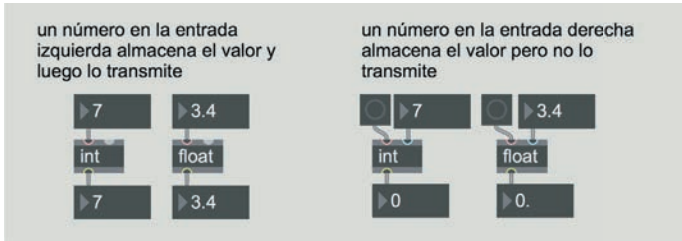


fig. IA.11: objetos `int` y `float`

...

© Contemponet 2020

El capítulo continúa con:

IA.2 GENERACIÓN DE NÚMEROS ALEATORIOS

IA.3 MANEJO DEL TIEMPO: EL OBJETO METRO

IA.4 SUBPATCHES Y ABSTRACCIONES
Los subpatches
Las abstracciones

IA.5 OTROS GENERADORES DE NÚMEROS ALEATORIOS

IA.6 ORDEN DE MENSAJES CON TRIGGER

IA.7 OBJETOS PARA ADMINISTRAR LISTAS
Los objetos unpack y unjoin
Los objetos pack y join
El objeto zl
Los objetos append y prepend
El objeto multislider

IA.8 LA MESSAGE BOX Y ARGUMENTOS VARIABLES
Comandos para objetos message box
Argumentos variables: el signo del dolar ($)
Argumentos variables: setdomain

IA.9 ENVÍO DE SECUENCIAS DE BANGS: EL OBJETO UZI

IA.10 SEND Y RECEIVE

IA.11 USO DE ATRIBUTOS DENTRO DEL OBJETO

IA.12 AUDIO MULTICANAL

ACTIVIDADES
• Sustitución de partes de algoritmos, corrección, compleción y análisis de algoritmos, construcción de nuevos algoritmos.

EVALUACIÓN
• Proyecto integrado: ingeniería en reversa

MATERIALES DE APOYO
• Lista de objetos Max - Lista de mensajes, atributos y parámetros de objetos Max específicos - Glosario

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

2T
SÍNTESIS ADITIVA Y SÍNTESIS VECTORIAL

2.1 SÍNTESIS ADITIVA DE ESPECTRO FIJO
2.2 BATIMIENTOS
2.3 FUNDIDOS CRUZADOS DE TABLAS DE ONDAS: SÍNTESIS VECTORIAL
2.4 SÍNTESIS ADITIVA DE ESPECTRO VARIABLE

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

AGENDA DE APRENDIZAJE

PRERREQUISITOS DEL CAPÍTULO

- Contenidos del capítulo 1 (teoría)

OBJETIVOS

Conocimientos

- Conocer la teoría de la suma de ondas (fase, interferencia constructiva, interferencia destructiva, etc.)
- Conocer la teoría y el uso básico de la síntesis aditiva de espectro fijo y espectro variable, de espectro armónico e inarmónico.
- Conocer la relación entre fase y pulsaciones (o batimiento).
- Conocer el modo en el cual se utilizan las tablas de ondas y la manera como ocurre la interpolación.
- Conocer la teoría y el uso básico de la síntesis vectorial

Habilidades

- Saber identificar auditivamente las características básicas de los sonidos armónicos e inarmónicos
- Saber reconocer auditivamente los batimientos
- Saber identificar las diferentes fases de la envolvente de un sonido y ser capaz de describir sus características

CONTENIDOS

- Síntesis aditiva de espectro fijo y variable
- Sonidos armónicos e inarmónicos
- Fase y batimientos
- Interpolación
- Síntesis vectorial

TIEMPO DE DEDICACIÓN - Capítulo 2 (Teoría y Práctica)

Autodidactas

Sobre un total de 300 horas de estudio individual (volumen I, teoría y práctica):

- Aproximadamente 60 horas

Cursos

Sobre un total de 60 horas de clase + 120 de estudio individual (volumen I, teoría y práctica):

- Aproximadamente 10 horas de clase frontal + 2 de "feedback"
- Aproximadamente 24 horas de estudio individual

ACTIVIDADES

Ejemplos interactivos

EVALUACIÓN

Prueba de preguntas con respuestas cortas
Prueba de escucha y análisis

MATERIALES DE APOYO

Conceptos fundamentales - Glosario - Discografía

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

2.1 SÍNTESIS ADITIVA DE ESPECTRO FIJO

El sonido producido por un instrumento acústico, o el sonido en general, es una oscilación compleja, expresable a través de un conjunto de vibraciones elementales que son producidas simultáneamente por el instrumento mismo: la suma de todas estas vibraciones contribuye de manera determinante al resultado sonoro que percibimos como timbre, y en consecuencia, determina la forma de onda. Cada sonido, por lo tanto, puede ser descompuesto en sinusoides que, tal como se dijo en el capítulo anterior, constituyen la pieza fundamental con la cual es posible construir cualquier otra forma de onda. Cada una de estas sinusoides, o componentes de sonido, tiene su propia frecuencia, amplitud y fase. El conjunto de frecuencias, amplitudes y fases de las sinusoides que forman un determinado sonido se define como **espectro sonoro**: más adelante veremos cómo puede ser representado gráficamente.
Tanto los sonidos naturales como los sintéticos pueden ser descompuestos en una serie de sinusoides: cada una de las formas de onda que hemos descrito en la sección 1.2, por lo tanto, tienen un espectro sonoro diferente, es decir, cada una contiene una mezcla (combinación) diferente de sinusoides (excepto la forma de onda sinusoidal, obviamente, ¡que se contiene solamente a sí misma!).

ESPECTRO Y FORMA DE ONDA

Espectro y forma de onda son dos términos usados para describir un sonido de dos maneras diferentes. La forma de onda es la representación gráfica de la amplitud en función del tiempo.[1] En el gráfico de la figura 2.1 podemos observar la forma de onda de un **sonido complejo**: en el eje x se encuentra el tiempo y en el eje y la amplitud; notemos que la forma de onda de este sonido es bipolar, esto es, sus valores de amplitud oscilan por encima y por debajo de cero. Esta es una representación gráfica en el **dominio del tiempo** (*time domain*), lo que significa que todos los valores de amplitud instantánea que forman la forma de onda del sonido complejo, instante por instante, son registrados.

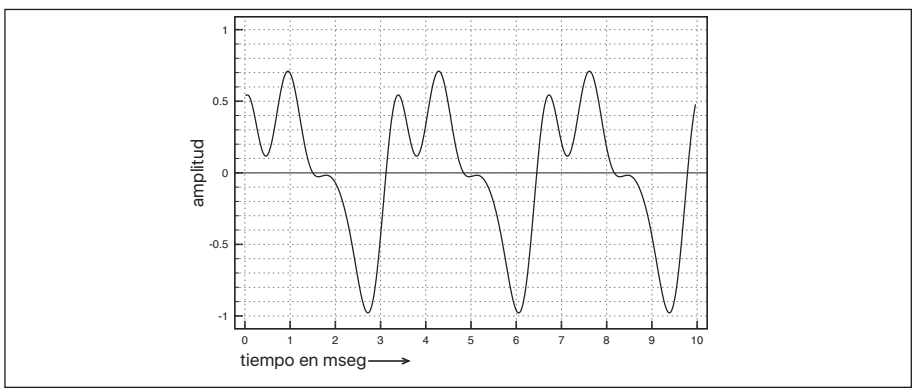


fig. 2.1: forma de onda de un sonido complejo

[1] En el caso de sonidos periódicos, la forma de onda puede ser representada por un único ciclo.

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

En la figura 2.2a en cambio, vemos cómo se puede descomponer en sinusoides el sonido complejo que acabamos de observar: hay cuatro sinusoides con valores de frecuencia y amplitud distintos que, al sumarse, constituyen el sonido complejo representado en el gráfico anterior.

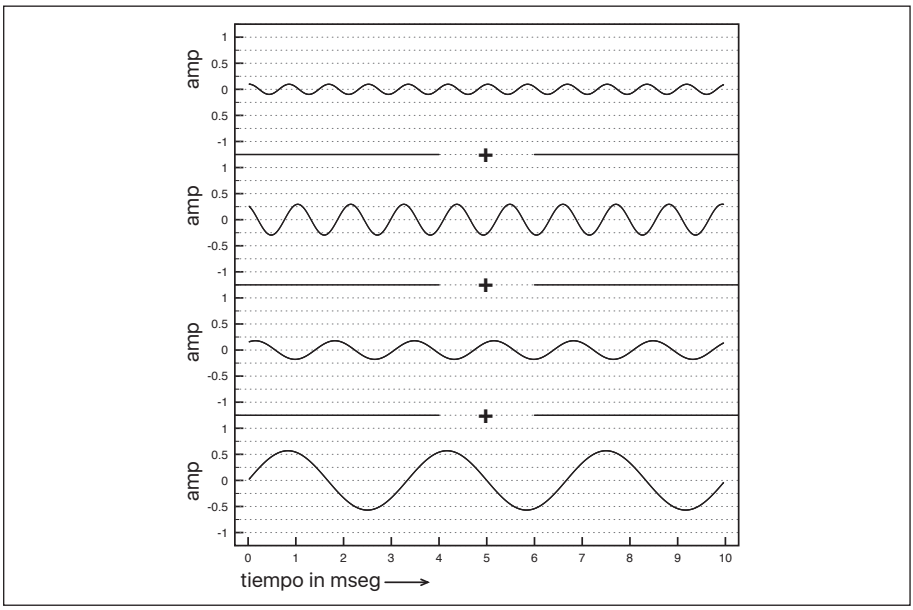


fig. 2.2a: descomposición en sinusoides de un sonido complejo

Para visualizar de manera más clara las diferentes frecuencias de este sonido y su amplitud, es posible recurrir a un gráfico de espectro sonoro, donde se representa la amplitud de sus componentes en función de la frecuencia (en el **dominio de la frecuencia** o *frequency domain*). En este caso, en el eje *x* se encuentran los valores de frecuencia, y en el eje y los valores de amplitud. La figura 2.2b muestra la amplitud de pico de cada componente presente en la señal.

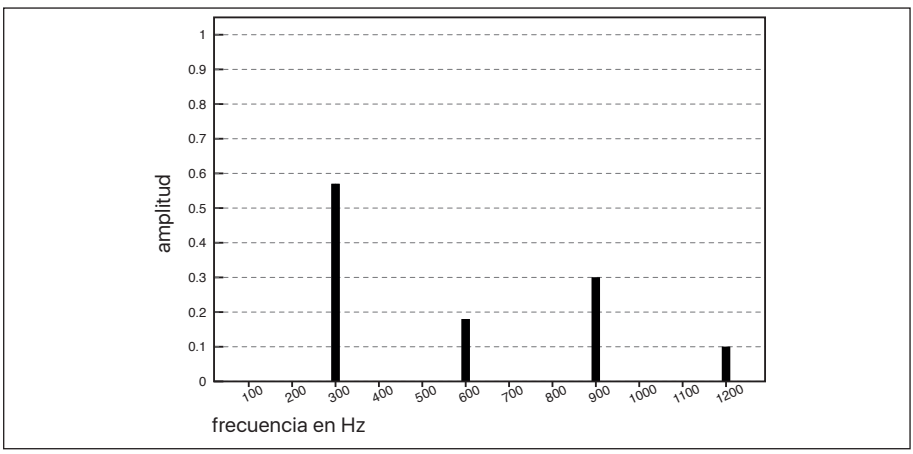


fig. 2.2b: espectro sonoro

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

Para observar el comportamiento de las componentes de un sonido en el tiempo se puede recurrir también al **sonograma**, que nos muestra en el eje y las frecuencias, y en el eje x el tiempo. (fig. 2.2c). Las líneas correspondientes a las frecuencias de las componentes están más o menos marcadas en función de su amplitud. En este caso hay solo 4 líneas rectas, porque se trata de un espectro fijo.

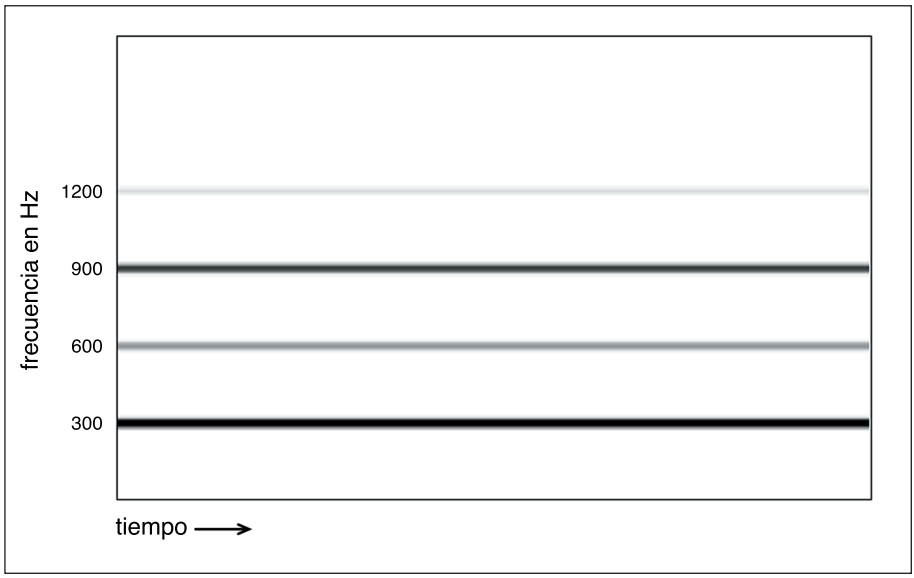


fig. 2.2c: sonograma (o espectrograma)

Ahora consideremos el proceso opuesto: en lugar de descomponer un sonido complejo en sinusoides, tomemos sinusoides aisladas para crear un sonido complejo. La técnica que permite crear cualquier forma de onda a partir de la suma de sinusoides se llama **síntesis aditiva** (fig. 2.3).

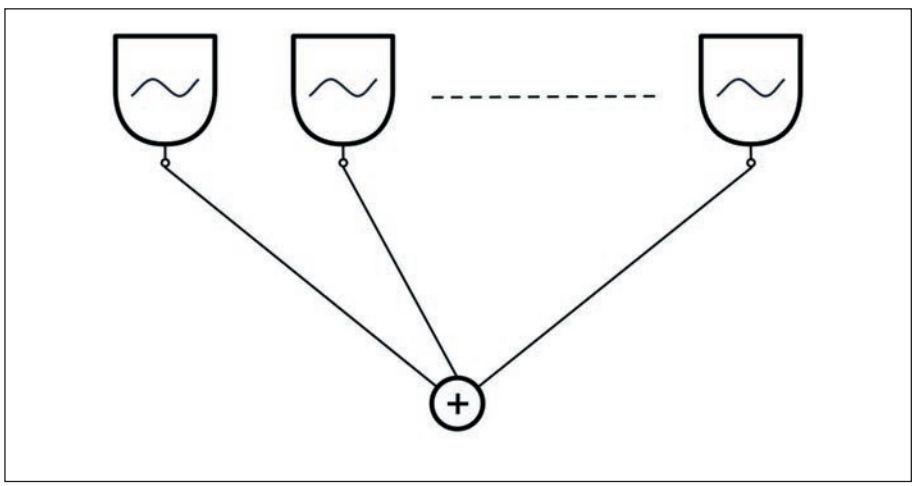


fig. 2.3: suma de señales en salida de osciladores sinusoidales

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

En la fig. 2.4. es posible ver dos ondas sonoras, A y B, y la respectiva suma, C.

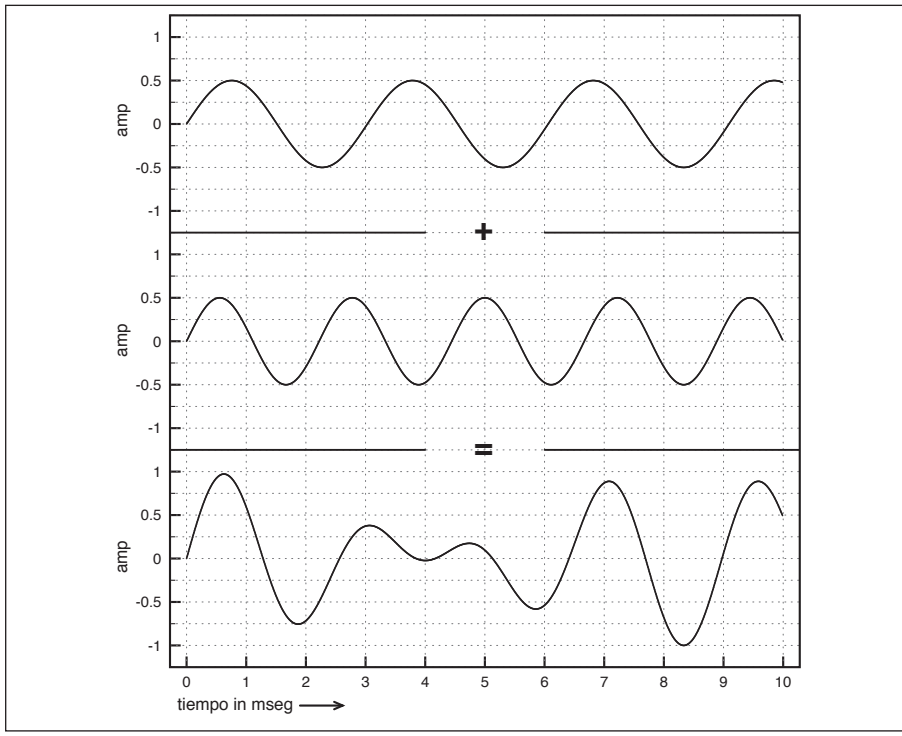


fig. 2.4: representación gráfica de una suma de sinusoides

Podemos verificar fácilmente que los valores de amplitud instantánea de la onda C se han obtenido sumando, punto por punto, los valores de amplitud instantánea de la onda A a los de la onda B. Instante por instante, los valores de amplitud de diferentes ondas se suman algebraicamente, es decir, con su respectivo signo, positivo o negativo. Si en un determinado instante, tanto la amplitud de A como la de B son positivas o negativas, el valor absoluto de la amplitud de C resultará mayor que cada uno de sus componentes, obteniendo de este modo una **interferencia constructiva**; veamos el siguiente ejemplo:

A = -0.3
B = -0.2
C = -0.5

Por el contrario, si en un determinado instante los valores de amplitud de A y B son uno positivo y el otro negativo, el valor absoluto de la respectiva suma algebraica será menor, al menos, de uno de los dos componentes, obteniendo de esta forma una **interferencia destructiva**; veamos el siguiente ejemplo:

A = 0.3
B = -0.1
C = 0.2

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

"La mayor parte, por no decir, casi todos los sonidos que escuchamos en el mundo real, está compuesta por sonidos complejos y no por sonidos puros. Los **sonidos complejos** pueden ser descompuestos en una mayor o menor cantidad de sonidos *puros*, y estos últimos a su vez, son llamados *componentes* del sonido complejo. Para comprender mejor este fenómeno, hagamos una analogía con la óptica. Se sabe que algunos colores son *puros*, lo que significa que no se pueden descomponer en otros (por ejemplo rojo, anaranjado, amarillo, hasta el color violeta). A cada uno de ellos le corresponde una determinada *longitud de onda* del rayo luminoso. En caso de estar presente solo uno de los colores puros, el prisma, que descompone la luz blanca en los siete colores del espectro luminoso, mostrará solamente ese componente. Lo mismo ocurre con el sonido. A una determinada **longitud de onda**[2] del sonido le corresponde una determinada altura percibida. Si no está presente simultáneamente ninguna otra frecuencia, el sonido será *puro*. Un sonido puro, como ya sabemos, tiene una forma de onda *sinusoidal*." (Bianchini, R., Cipriani, A., 2001, pp. 69-70)

Las componentes de un sonido complejo pueden tener frecuencias que son múltiplos enteros de la frecuencia de la componente más grave. En este caso, la componente más grave se llama **fundamental** y las otras se llaman **componentes armónicas** (ejemplo: si la fundamental es igual a 100 Hz, las otras componentes armónicas tendrán las siguientes frecuencias: 200 Hz, 300 Hz, 400 Hz, etc.).
La componente que tiene una frecuencia doble respecto a la de la fundamental, recibe el nombre de *segundo armónico*; la componente de frecuencia triple respecto a la fundamental, es llamada *tercer armónico*, etc. Cuando las componentes del sonido son múltiplos enteros de la fundamental (como en este caso), se dice que el sonido es armónico. Notemos que en un sonido armónico, la frecuencia de la fundamental representa el *máximo común divisor* de las frecuencias de todas las otras componentes: es decir, el número máximo que divide exactamente (sin decimales) todas las frecuencias.

EJEMPLO INTERACTIVO 2A • *COMPONENTES ARMÓNICAS*

Si los sonidos puros que componen un sonido complejo no son múltiplos enteros de la componente más grave, estamos ante un sonido inarmónico, y las componentes reciben el nombre de …

…

[2] "La longitud de un ciclo se define como **longitud de onda** y se mide en metros [m] o [cm]. Este es el espacio que un ciclo ocupa físicamente en el aire, y si el sonido fuera visible, podría ser medido fácilmente, por ejemplo con un metro." (Bianchini, R. 2003)

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

El capítulo continúa con:

La fase
Espectros armónicos e inarmónicos
Periódico/aperiódico, armónico/inarmónico
Interpolación en la lectura de tablas

2.2 BATIMIENTOS

2.3 FUNDIDOS CRUZADOS DE TABLAS DE ONDAS: SÍNTESIS VECTORIAL

2.4 SÍNTESIS ADITIVA DE ESPECTRO VARIABLE

ACTIVIDADES
• Ejemplos interactivos

EVALUACIÓN
• Prueba de preguntas con respuestas cortas prueba de escucha y análisis

MATERIALES DE APOYO
• Conceptos fundamentales - Glosario - Discografía

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

2P
SÍNTESIS ADITIVA Y SÍNTESIS VECTORIAL

2.1 SÍNTESIS ADITIVA DE ESPECTRO FIJO
2.2 BATIMIENTOS
2.3 FUNDIDOS CRUZADOS ENTRE TABLAS DE ONDAS: SÍNTESIS VECTORIAL
2.4 SÍNTESIS ADITIVA DE ESPECTRO VARIABLE

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

AGENDA DE APRENDIZAJE

PRERREQUISITOS DEL CAPÍTULO

- Contenidos del Capítulo 1 (Teoría y Práctica), Capítulo 2 (Teoría), Interludio A

OBJETIVOS

Habilidades

- Saber sintetizar un sonido complejo a partir de ondas sinusoidales simples
- Saber sintetizar sonidos armónicos e inarmónicos usando síntesis aditiva y tablas de onda, y transformar los unos en los otros y viceversa, usando controles de amplitud y de frecuencia
- Saber implementar formas de onda triangulares, cuadradas y diente de sierra de forma aproximada por medio de la adición de ondas sinusoidales armónicas
- Saber controlar batimientos entre dos sonidos sinusoidales o armónicos
- Saber sintetizar sonidos usando síntesis vectorial

Competencia

- Saber realizar un estudio sonoro basado en la técnica de síntesis aditiva y grabarlo en un archivo de audio

CONTENIDOS

- Síntesis aditiva usando tanto espectros fijos como variables
- Sonidos armónicos e inarmónicos
- Fase y batimientos
- Interpolación
- Síntesis vectorial

TIEMPO DE DEDICACIÓN - Capítulo 2 (Teoría y Práctica)

Autodidactas

Sobre un total de 300 horas de estudio individual (volumen I, teoría y práctica):

- Aproximadamente 60 horas

CURSOS

Sobre un total de 60 horas de clase + 120 de estudio individual (volumen I, teoría y práctica):

- Aproximadamente 10 horas de clase frontal + 2 de "feedback"
- Aproximadamente 24 horas de estudio individual

ACTIVIDADES

- Actividades en la computadora: sustitución de partes de algoritmos, corrección, completación y análisis de algoritmos, construcción de nuevos algoritmos.

EVALUACIÓN

- Realización de un estudio breve.
- Proyecto integrado de Ingeniería en Reversa

MATERIALES DE APOYO

- Lista de objetos de Max - Lista de mensajes, atributos y parámetros de objetos de Max específicos - Glosario

© Contemponet 2020

2.1 SÍNTESIS ADITIVA DE ESPECTRO FIJO

Para empezar, vamos a crear un *patch* para producir sonidos armónicos en Max usando síntesis aditiva, a partir de la figura 2.12 del capítulo teórico como guía de implementación. El diagrama en esa figura muestra 10 osciladores cuyas salidas se suman usando un mezclador, así que empezaremos con 10 objetos **`cycle~`** que proporcionarán las 10 ondas sinusoidales, cada una con una frecuencia que será un múltiplo entero de la fundamental. Para calcular los valores de frecuencia necesarios, simplemente multiplicaremos la frecuencia de la fundamental por los primeros 10 números enteros. El *patch* resultante es mostrado en la figura 2.1 (**02_01a_aditiva.maxpat**).

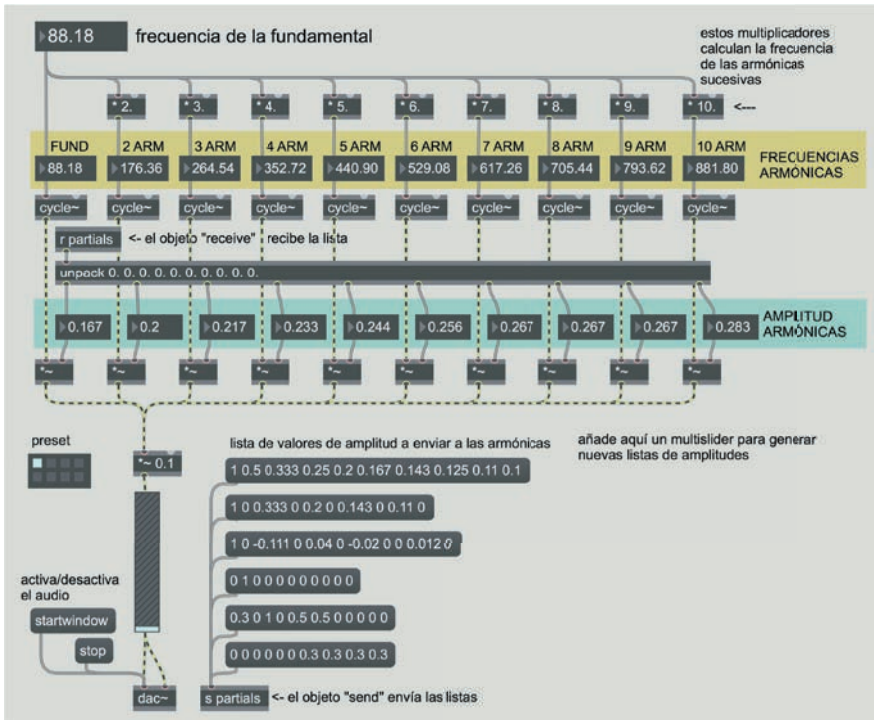


fig. 2.1: archivo **02_01a_aditiva.maxpat**

La *number box* de la parte superior del *patch* es usada para definir la frecuencia fundamental del *patch* completo, posteriormente, esta es enviada a los 9 operadores **`*~`** , los cuales la multiplican por números enteros sucesivos para producir frecuencias armónicas que sean múltiplos de la fundamental. Tanto la frecuencia de la fundamental como las producidas por estos multiplicadores pueden ser observadas en las 10 *number box* de la parte inferior, que a su vez están conectadas a los correspondientes objetos **`cycle~`**. Estos también están conectados a otros tantos multiplicadores de señal (*~) que reescalan sus salidas. Normalmente, la amplitud de una señal producida por **`cycle~`** tendrá un valor máximo de 1, y por esta razón, el valor dado a cada multiplicador sirve para reflejar directamente la amplitud relativa de cada armónica (por ejemplo: 1.0 * 0.5 = 0.5).

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

Los multiplicadores de señal permiten que cada armónica tenga su propia amplitud distintiva, determinada por el valor alimentado a su multiplicador. Dado que tenemos 10 osciladores, 10 valores especifican completamente las amplitudes a ser usadas.

Como aprendimos en la sección IA.10 del Interludio A, podemos crear una lista de números usando una *message box* o un `multislider`.
Veamos el primer caso: las amplitudes deseadas son agrupadas en listas en el interior de *message box*, y luego son despachadas a los 10 multiplicadores usando un objeto `send` [`s` partials]. El objeto `receive` correspondiente es conectado a un objeto `unpack`, que se encarga de distribuir los valores, uno por uno, a los multiplicadores apropiados (como se observa en la figura 2.1b).

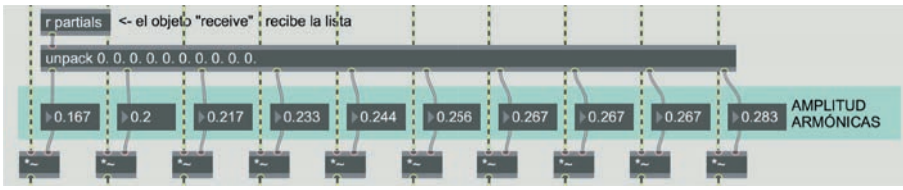


fig. 2.1b: el objeto `unpack` convierte una lista en elementos unitarios

Sin embargo, las listas de amplitudes también podrían ser creadas con un `multislider`. Crea uno ahora en la sección inferior derecha del *patch* y configúralo usando el *Inspector* de manera que tenga 10 *sliders*, cada uno con un rango entre 0 y 1.
Este `multislider` necesitará ser conectado a un objeto `send` con "partials" como argumento. También es posible manipular las *float number box* conectadas a la entrada derecha de los multiplicadores de señal para definir así diferentes amplitudes, y luego grabar el resultado usando el objeto `preset` de la izquierda (haciendo *shift-click* en una de sus casillas).
Las señales que componen este *patch*, tras ser reescaladas por sus multiplicadores individuales de amplitud, son ruteadas a un objeto final de multiplicación que reduce la amplitud total de la señal sumada a un 1/10 de la amplitud original. Sabemos que dos o más señales alimentadas a una misma entrada se suman. Tratándose de 10 osciladores, por tanto, si todos los componentes tienen una amplitud máxima de 1, la forma de onda resultante de la suma tendría una amplitud máxima de 10. Este multiplicador, por consiguiente, sirve para volver el valor absoluto de la señal a un valor entre 0 y 1, evitando así la distorsión del sonido.[1]

El `multislider` de ampltiud debe despachar datos continuamente, incluso cuando el valor de un *slider* está siendo modificado: todas las variaciones deben ser enviadas al multiplicador a medida que estas van ocurriendo. Sin embargo, por defecto `multislider` actualiza la lista de valores de su salida solo cuando se suelta la tecla del ratón, al final de una edición.

[1] Para más información sobre distorsión debido a amplitud excesiva, revisar la sección 1.2 del capítulo 1 de la parte teórica.

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

Para configurar el objeto para que produzca actualizaciones de forma continua, debes usar el *inspector* del objeto y seleccionar "Continous Data Output When Mousing" (en la categoría "Style").

Ya hemos dicho que gracias a la posibilidad de manejar canales audio múltiples, es muy fácil crear *patches* en los cuales varias señales sean procesadas en paralelo: el *patch* de la figura 2.1 todavía no goza de esta posibilidad porque ante todo, hemos querido mostrar el algoritmo completo para que se pueda entender mejor la ruta de las señales individuales.
Ahora veremos cómo se puede realizar el mismo algoritmo utilizando señales multicanal: para ello, carga el patch **02_01b_aditiva_multicanal.maxpat** (fig. 2.1c)

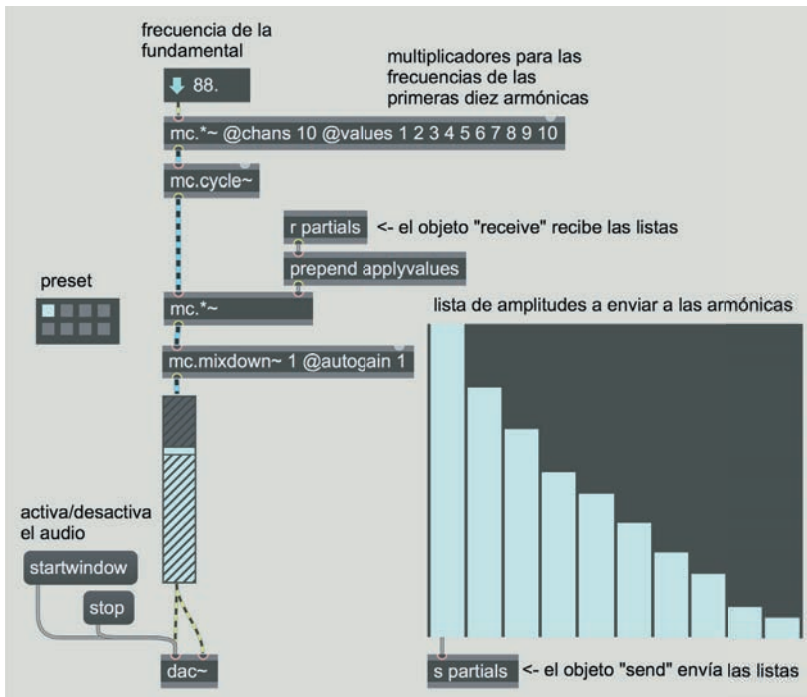


fig. 2.1c: archivo **02_01b_aditiva_multicanal.maxpat**

Analicemos el *patch* iniciando por la parte superior: el primer objeto es un `number~` en modalidad "generador de señal"; observa que la pequeña onda que normalmente aparece en la parte izquierda ha sido sustituida por una flecha que apunta hacia abajo. En esta modalidad, el objeto `number~` genera una señal constante que corresponde al valor mostrado. Para pasar de una modalidad a la otra, es necesario hacer click sobre el ícono (onda o flecha) presente a la izquierda del objeto. El valor generado por `number~` (88, en la figura) es pasado al multiplicador multicanal `mc.*~`: observa que en el interior de este último objeto ha sido configurado, como primer atributo, @chans 10". Es fácil darse cuenta de que este atributo determina el número de canales audio que deseamos (es decir, el número de copias o instancias del multiplicador de señales): la señal (monofónica) transmitida por el objeto `number~` es enviada a todos los canales del objeto `mc.*~`.

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

El segundo atributo es "@values 1 2 3 4 5 6 7 8 9 10" que configura los factores de multiplicación por cada instancia (ver también la figura IA.74b en la sección IA.12). Los valores generados por el multiplicador son pasados al oscilador múltiplo **mc.cycle~** que, al recibir una señal de 10 canales, genera 10 osciladores sinusoidales (en consecuencia, en este caso no hay necesidad del atributo @chans, porque el objeto **mc.cycle~** se adapta al número de canales que recibe). La amplitud de cada oscilador posteriormente es reescalada por otro multiplicador **mc.*~** que recibe, a través de la pareja de objetos [**s** partials] y [**r** partials], la lista de valores de amplitud generada por **multislider**: antes se pasar al multiplicador, a la lista se le propone (a través de prepend) el mensaje "applyvalues", con el cual cada valor es asignado a un canal diferente (ver también la figura IA.74c en la sección IA.12).

Posteriormente, el objeto **mc.mixdown~** reduce la señal múltiple resultante a una señal individual: el atributo "@autogain 1", como ya sabemos, reescala la señal de tal manera que la amplitud instantánea no supere los valores máximos -1/1 (este atributo en esta sede desempeña la misma función realizada por el multiplicador [*~ 0.1] en el *patch* de la figura 2.1).

Veamos ahora una característica del sistema multicanal que nos permite simplificar adicionalmente el *patch*: carga el archivo **02_01c_aditiva_harmonic.maxpat** (fig. 2.1d).

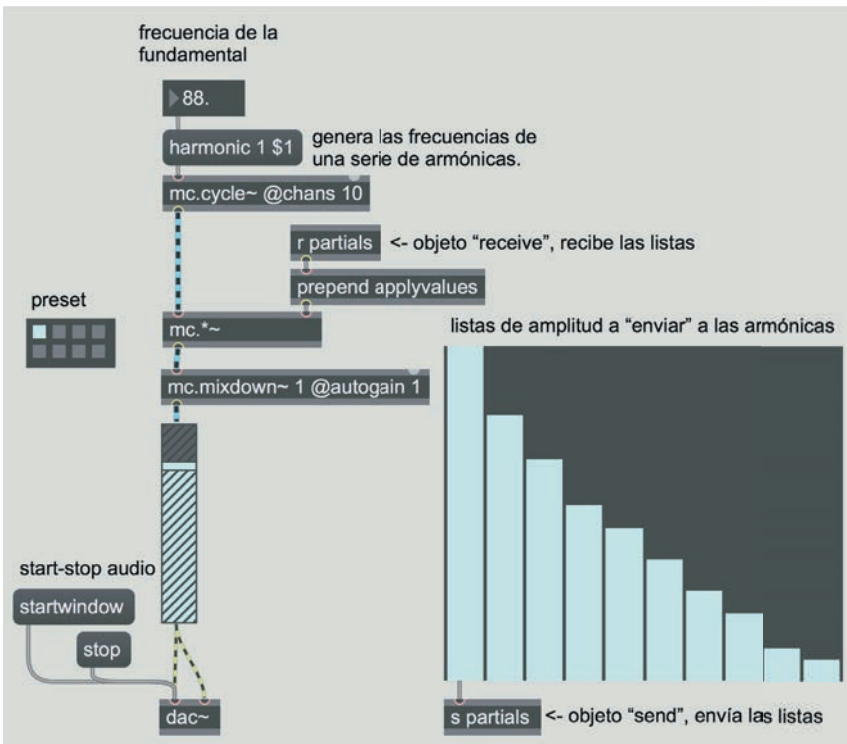


fig. 2.1d: *patch* **02_01c_aditiva_harmonic.maxpat**

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

Como habrás notado, hemos sustituido el multiplicador presente en la parte superior del *patch* anterior con el mensaje [harmonic 1$1]. Este mensaje es parte de una serie de mensajes "generadores" que configuran, según determinadas funciones, los valores de las instancias contenidas en un *wrapper* multicanal[2]: "harmonic" es el nombre del generador, el valor sucesivo, 1, indica una secuencia normal de armónicas (más adelante veremos qué es lo que se genera con valores diferentes) y el último valor determina la frecuencia fundamental de la serie armónica. La serie generada en el *patch* de la figura será, por lo tanto, 88, 176, 264, etc. Cada uno de estos valores es enviado a un oscilador diferente. Este es solo uno de los mensajes que tenemos a disposición para generar valores en el sistema multicanal: veremos otros dentro de poco.
Observa que hemos añadido al objeto `mc.cycle~` el atributo "@chans 10" (en el *patch* anterior este se encontraba en el multiplicador que antes se encontraba en la parte superior y que luego fue cancelado), así el *wrapper* sabrá cuántas copias (o instancias) del oscilador deseamos.

El *patch* de la figura 2.1d es mucho más "ligero" respecto al *patch* equivalente de la figura 2.1, y nos permite comprender la utilidad del sistema multicanal. No obstante, no hay que pensar que el segundo *patch*, aunque tenga menos objetos, sea más liviano para el CPU: en realidad, como ya dijimos, el *wrapper* multicanal crea el número de objetos necesarios para manejar todas las señales; esto significa que también en el *patch* de la figura 2.1d, aunque no se vean, hay 10 osciladores `cycle~` y 10 multiplicadores MSP.
Utiliza este *patch* para generar varios perfiles de espectro; grábalos como presets y luego trata de comparar las diferencias tímbricas que se crean al enfatizar las componentes graves, medias y agudas.

ACTIVIDADES

Usando el *patch* **02_01c_aditiva_hamonic.maxpat**, crea dos presets, uno con las ampltiudes de las armónicas impares configuradas en 1 y las de los pares en 0. El segundo *preset* debe tener la amplitudes de las armónicas pares configuradas en 1 y las de los impares en 0. Además del timbre, ¿cuál es la diferencia más evidente? ¿Puedes explicarlo?

Continuando con el uso de **02_01c_aditiva_hamonic.maxpat** (en referencia al tema de las fundamentales ausentes de la sección 2.1 de la teoría), utiliza un espectro en el que todas las amplitudes de las armónicas sean 1, y luego reduce a cero la amplitud de la fundamental. Escucha la ilusión de la fundamental que hemos puesto a cero y que parece estar presente. Luego escucha lo que pasa mientras reduces la amplitud de las armónicas sucesivas, una por una. ¿En qué momento ya no se percibe la fundamental?

[2] Recuerda que el *wrapper* es el "embalaje" que envuelve las instancias (copias) de los objetos MSP necesarios para generar la señal múltiple: revisa la sección IA.12

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

LA FASE

Ahora abordaremos un tema discutido en profundidad en la sección 2.1 del capítulo 2 de la teoría: la fase de una forma de onda periódica. Es un concepto básico que será desarrollado a lo largo de muchos capítulos, por eso es importante entender cómo se maneja la fase en Max. Probablemente habrás notado que el objeto **`cycle~`** tiene dos entradas: la izquierda, como ya sabemos, nos permite definir la frecuencia, y la derecha nos permite definir la fase normalizada, una fase que es expresada como un valor entre 0 y 1 (en vez de un valor entre 0 y 360 grados o entre 0 y 2π radianes, ambos estudiados en el recuadro dedicado a la fase, en la sección 2.1T).
Construye el *patch* mostrado en la figura 2.2 en un nuevo *Patcher Window*.

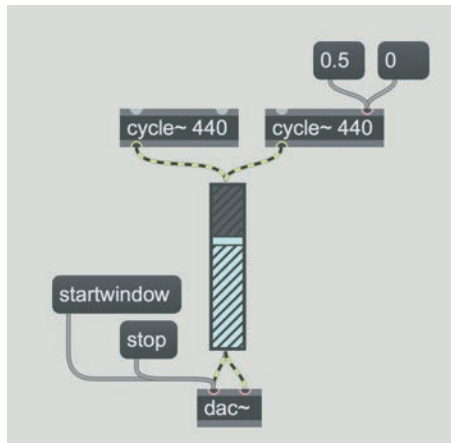


fig. 2.2: interferencia constructiva y destructiva

Tenemos dos objetos **`cycle~`** a 440 Hz, el segundo tiene dos *message box* que contienen los números 0.5 y 0, y están conectados a su entrada derecha (su entrada de fase, como acabamos de ver). Enciende el *patch* haciendo click en "startwindow" y sube el fader de **`gain~`**: escucharás un La a 440 Hz perfectamente en fase, generado por los dos osciladores.

Haciendo click en la *message box* que contiene el valor 0.5, el sonido desaparece. ¿Sabes por qué? Hemos definido la fase del segundo oscilador a 0.5, lo cual corresponde a medio ciclo (180 grados). Los dos osciladores quedan con polaridad inversa[3] y por esto, su suma es siempre cero. Haciendo click en la *message box* que contiene el número 0, volveremos a hacer que los dos osciladores estén en fase y escucharemos el sonido una vez más.

Para entender mejor las combinaciones de fase, modifica el *patch* según la figura 2.3.

[3] Ver el recuadro dedicado a fase en la sección 2.1 de la parte teórica.

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

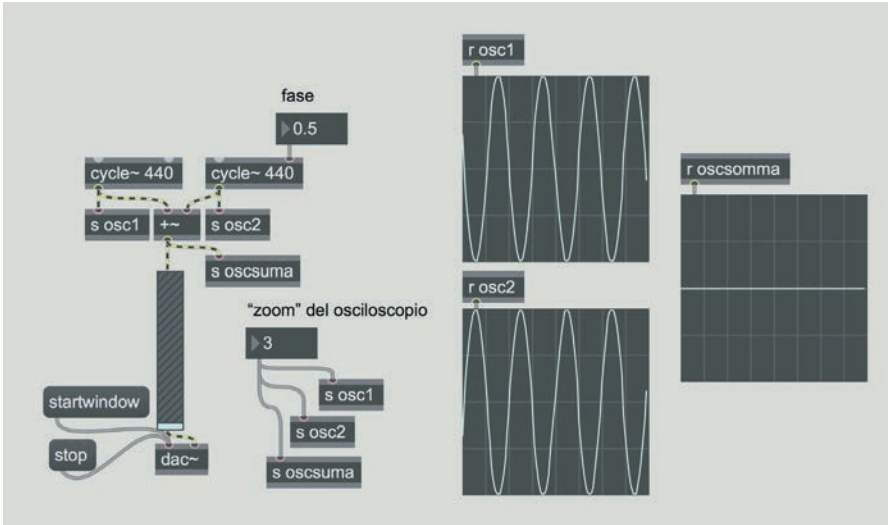


fig. 2.3: variaciones de fase

Ahora hemos reemplazado las dos *message box* conectadas a la entrada derecha del segundo oscilador con una *float number box*, y hemos añadido tres objetos **send** para rutear las señales producidas por los osciladores y su suma a tres objetos **scope~**. Además, podemos enviar el atributo "Samples per Pixel"[4] a los tres osciloscopios para controlar su apariencia. Esto se hace usando tres objetos **send** adicionales, todos conectados a una *number box* para números enteros. Notarás que tanto valores de señal como numéricos son enviados a los osciloscopios usando el mismo nombre ("osc1", "osc2" y "oscsuma") - recuerda que el objeto **scope~** acepta en la misma entrada tanto las señales a ser mostradas como valores de parámetros para controlar la visualización.

Después de configurar las *number box* a los valores mostrados en al figura y hacer click en "startwindow", deberías de ver ondas sinusoidales de polaridad inversa en los primeros dos osciloscopios, y una forma de onda "nula" (una línea plana) en el tercero. Haz click en "stop" y comprueba que las dos ondas sinusoidales realmente tengan una fase opuesta (como se ve en la figura). Ahora, tras hacer click nuevamente en "startwindow", modifica el valor en la *float number box* para controlar la fase del segundo oscilador. Cuando empieza a caer desde 0.5 hacia 0, los osciladores ya no se encuentran en fase opuesta, y se escucha un sonido creciendo progresivamente en amplitud hasta que los dos osciladores están completamente en fase y sus rangos de amplitud sumada están entre -2 y 2.[5] Prueba usando valores de fase mayores a 1, y observa que cada vez que la fase es igual a un entero,

[4] Ver la sección 1.2, en la cual demostramos que este atributo puede ser considerado, aunque de manera algo inapropiada, como un tipo de "factor de zoom" del osciloscopio.

[5] La onda resultante oscila entre -2 y 2 porque es la suma de dos ondas coseno (ya lo veremos más adelante) que oscilan entre -1 y 1 y están perfectamente en fase.

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

el sonido llega a su amplitud máxima, y cada vez que la fase está exactamente a medio camino entre dos valores enteros (como 1.5 - 2.5, - 3.5, etc.) el sonido es anulado. Los valores de fase superiores a 1 (mayores que 360 grados) hacen que el ciclo dé la vuelta comenzando desde el principio, tal como explicamos en el recuadro dedicado a la fase, en la sección 2.1 de la teoría. Por supuesto, todo esto funciona solo si los dos generadores comparten exactamente la misma frecuencia. Si no lo hacen, la razón entre fases de ambos osciladores variará constantemente, dando origen, en caso de que dos frecuencias sean solo levemente diferentes (por ejemplo 440 y 441 Hz), al fenómeno de batimientos, que será tratado en la sección 2.2.

Observemos otro posible uso para la entrada derecha de `cycle~` (fig. 2.4).

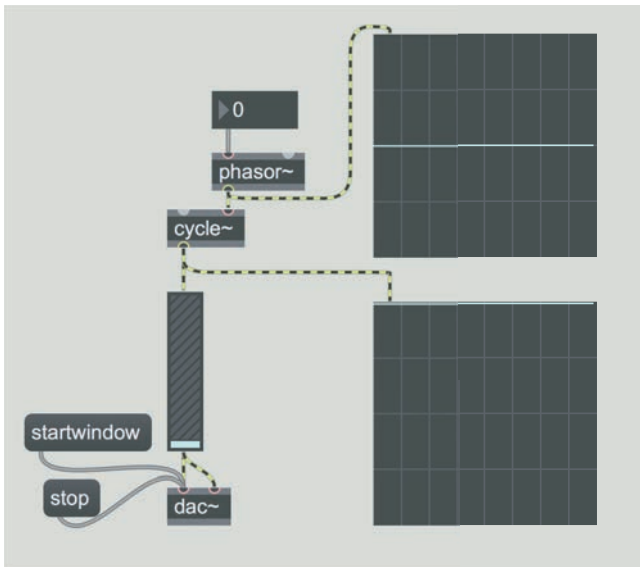


fig. 2.4: oscilador controlado por un `phasor~`

Aquí tenemos un `cycle~` configurado en 0 Hz (puesto que no tiene ni un argumento ni ha recibido un valor numérico en su entrada izquierda), cuya fase es modificada por un `phasor~`, que en la figura también tiene una frecuencia de 0 Hz (y por lo tanto, no se mueve). Observando los osciloscopios, vemos que el `phasor~` genera un torrente de muestras de valor 0, lo que implica que está congelado en el comienzo de su ciclo, y `cycle~` genera un flujo de muestras de valor igual a 1, lo que significa que también está congelado en el comienzo de su ciclo. Es importante destacar que el ciclo de la forma de forma de onda generada por `cycle~` empieza con el valor 1: como hemos mencionado en la sección 1.1, `cycle~` genera una onda coseno, lo que explica por qué el ciclo empieza en 1, su valor positivo máximo.[6] Pero esto no es una regla general, `phasor~`, por ejemplo, empieza su ciclo en 0.

[6] Para una definición de la función coseno, ver las secciones teóricas 1.2 y 2.1.

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

Ahora le daremos a **phasor~** una frecuencia mayor a 0 Hz; lo configuraremos en 5 Hz, como en el ejemplo mostrado en la figura 2.5.

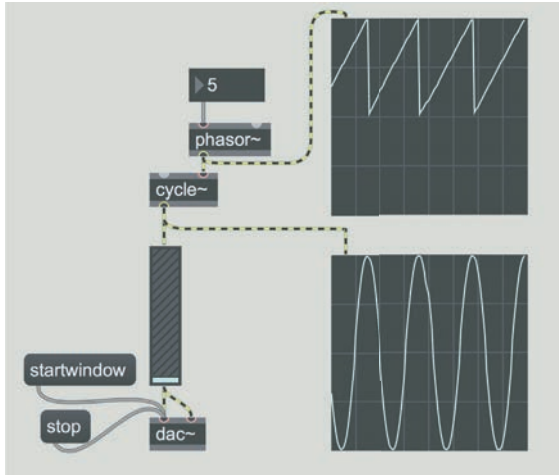


fig. 2.5: **phasor~** configurado a 5 Hz

Construye tú mismo este *patch*. La fase del objeto **cycle~** es esclava del objeto **phasor~**, y hace que oscile a la misma frecuencia y con la misma fase que **phasor~**. En la sección 1.2 vimos que **phasor~** genera una rampa de 0 a 1, y esta rampa, cuando es aplicada a la entrada de fase de **cycle~**, produce la oscilación (esto explica el motivo por el cual **phasor~** recibe este nombre: una de sus funciones principales es controlar la fase de otros objetos).

Como vemos en la figura 2.5, a cada rampa generada por **phasor~** le corresponde una oscilación completa de **cycle~**, que está corriendo a 5 Hz en este ejemplo. Si eliminásramos el **phasor~** y le diéramos a **cycle~** una frecuencia de 5 Hz a través de un mensaje en su entrada, o proporcionándole un argumento, obtendríamos una oscilación idéntica. En otras palabras, **cycle~** se comporta como si tuviera un "phasor interno".

...

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

El capítulo continúa con:

Uso de tablas de ondas con osciladores
Espectro fijo inarmónico

2.2 BATIMIENTOS
Batimientos rítmicos
Batimientos de componentes armónicos

2.3 FUNDIDOS CRUZADOS DE TABLAS DE ONDAS: SÍNTESIS VECTORIAL

2.4 SÍNTESIS ADITIVA DE ESPECTRO VARIABLE
Trabajo con envolventes de componentes usando una interfaz de usuario
Trabajo con envolventes de componentes únicos usando descripciones de texto
Uso de bancos de osciladores
Conversión de milisegundos a muestras, y viceversa
Espectros variables y bancos de osciladores
Uso de masking para controlar

ACTIVIDADES
• Actividades en la computadora: sustitución de partes de algoritmos, corrección, compleción y análisis de algoritmos, construcción de nuevos algoritmos

EVALUACIÓN
• Realización de un estudio breve.
• Proyecto integrado de ingeniería en reversa

MATERIALES DE APOYO
• Lista de objetos de Max - Lista de mensajes, Atributos y parámetros de objetos de Max específicos - Glosario

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

3T
GENERADORES DE RUIDO, FILTROS Y SÍNTESIS SUSTRACTIVA

3.1 FUENTES SONORAS PARA LA SÍNTESIS SUSTRACTIVA
3.2 FILTROS PASA BAJOS, PASA ALTOS, PASA BANDA Y RECHAZA BANDA
3.3 EL FACTOR Q O FACTOR DE RESONANCIA
3.4 ÓRDENES DE FILTROS Y CONEXIÓN EN SERIE
3.5 LA SÍNTESIS SUSTRACTIVA
3.6 ECUACIONES DE FILTROS DIGITALES
3.7 FILTROS CONECTADOS EN PARALELO, Y ECUALIZACIÓN GRÁFICA
3.8 OTRAS APLICACIONES DE CONEXIÓN EN SERIE: ECUALIZADORES PARAMÉTRICOS Y FILTROS SHELVING
3.9 OTRAS FUENTES PARA SÍNTESIS SUSTRACTIVA: IMPULSOS Y CUERPOS RESONANTES

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri

© Contemponet 2020

AGENDA DE APRENDIZAJE

PRERREQUISITOS DEL CAPÍTULO

- Contenidos de los capítulos 1 y 2

OBJETIVOS

Conocimientos

- Conocer la teoría de la síntesis sustractiva
- Conocer la teoría y el uso de los parámetros de los filtros principales
- Conocer la diferencia entre la teoría de los filtros ideales y la respuesta de los digitales
- Conocer la teoría y la respuesta de los filtros FIR e IIR
- Conocer el uso de los filtros dispuestos en serie o en paralelo
- Conocer la teoría y el uso de los ecualizadores gráficos y paramétricos
- Conocer la aplicación de los filtros a diferentes tipos de señal
- Conocer las funciones principales de un típico sintetizador de síntesis sustractiva

Habilidades

- Saber identificar auditivamente las características fundamentales de un filtrado y saber describirlas

CONTENIDOS

- Síntesis sustractiva
- Filtros pasa bajos, pasa altos, pasa banda y rechaza banda
- Filtros High Shelving, Low Shelving, Peak/Notch
- Factor Q
- Órdenes de los filtros
- Filtros de respuesta infinita al impulso y de respuesta finita al impulso
- Ecualizadores gráficos y paramétricos
- Filtrado de sonidos producidos por generadores de ruido, sonidos muestreados, impulsos

TIEMPO DE DEDICACIÓN - Capítulo 3 (Teoría y Práctica) + Interludio B

Autodidactas

Sobre un total de 300 horas de estudio individual (volumen I, teoría y práctica):

- Aproximadamente 110 horas

Cursos

Sobre un total de 60 horas de clase + 120 de estudio individual (volumen I, teoría y práctica):

- Aproximadamente 18 horas de clase frontal + 4 de "feedback"
- Aproximadamente 44 horas de estudio individual

ACTIVIDADES

- Ejemplos interactivos

EVALUACIÓN

- Prueba de preguntas de respuestas cortas
- Prueba de escucha y análisis

MATERIALES DE APOYO

- Conceptos fundamentales - Glosario - Discografía

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

En este capítulo hablaremos de *filtros*, argumento fundamental en el campo del diseño sonoro y de la música electrónica, y de una técnica de síntesis que los usa considerablemente: la síntesis sustractiva.

3.1 FUENTES SONORAS PARA LA SÍNTESIS SUSTRACTIVA

La **síntesis sustractiva** nace de la idea de poder crear un sonido modificando la amplitud de algunas componentes de otro sonido, a través del uso de filtros. La finalidad de la mayor parte de los filtros digitales, de hecho, es alterar de alguna manera el espectro sonoro. Un **filtro**, por lo tanto, es un dispositivo que actúa prevalentemente sobre algunas frecuencias contenidas en una señal, habitualmente atenuándolas o enfatizándolas.[1]
Cualquier sonido puede ser filtrado. ¡Pero cuidado! No podemos atenuar o enfatizar componentes que no existan en el sonido original, por ejemplo, no tendría sentido usar un filtro para enfatizar los 50 Hz si estamos filtrando una voz de soprano, simplemente porque esta frecuencia no está presente en el sonido original.
Los sonidos originarios que generalmente se utilizan en la síntesis sustractiva son ricos desde un punto de vista espectral, y tal como hemos dicho, los filtros se utilizan para modelar el espectro de estos sonidos y obtener, en la señal de salida, sonidos distintos respecto a los originarios.
En esta sección nos concentraremos en los sonidos típicos utilizados como fuente para la síntesis sustractiva y para el uso de filtros en general.
En las siguientes secciones abordaremos las técnicas de filtrado.

En general, cuando se trabaja con filtros en un estudio se utilizan diversos tipos de sonidos:

- Sonidos producidos por: generadores de ruido, generadores de impulsos, bancos de osciladores, otros generadores de señal y algoritmos de síntesis.
- Archivos de audio/sonidos muestreados.
- Sonidos producidos por fuentes en vivo en tiempo real (por ejemplo, un sonido proveniente del micrófono de un oboísta en acción).

GENERADORES DE RUIDO: RUIDO BLANCO Y RUIDO ROSADO

Uno de los sonidos más utilizados como fuente para la síntesis sustractiva es el **ruido blanco**, un sonido que contiene todas las frecuencias audibles y cuyo espectro es esencialmente plano (no obstante la amplitud de cada frecuencia esté distribuida aleatoriamente).
Es llamado ruido blanco por analogía con la óptica, dado que el color blanco contiene todos los colores del espectro visible. Se utiliza a menudo el ruido blanco como sonido fuente porque puede ser filtrado útilmente por cualquier tipo de filtro a cualquier frecuencia, dado que contiene todas las frecuencias audibles (ver fig. 3.1).

[1] Además de la amplitud, un filtro puede modificar la fase de las componentes de un sonido.

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

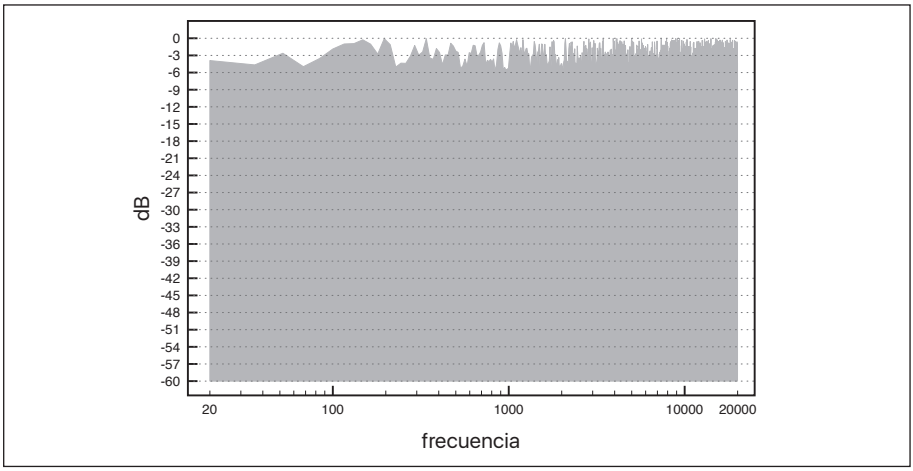


fig. 3.1: espectro de ruido blanco

Otro tipo de ruido utilizado a menudo en la síntesis sustractiva es el **ruido rosado**. Este, a diferencia del ruido blanco, tiene un espectro cuya energía disminuye con el aumento de la frecuencia, de forma más precisa, con una atenuación de 3 dB por octava[2]; también es llamado generador de ruido 1/f para indicar que su energía espectral es proporcional al recíproco de la frecuencia. A menudo es utilizado, junto a un analizador de espectro, para probar y corregir la respuesta en frecuencia de equipos audio con relación a ambientes en los cuales se realizan eventos musicales (fig. 3.2).

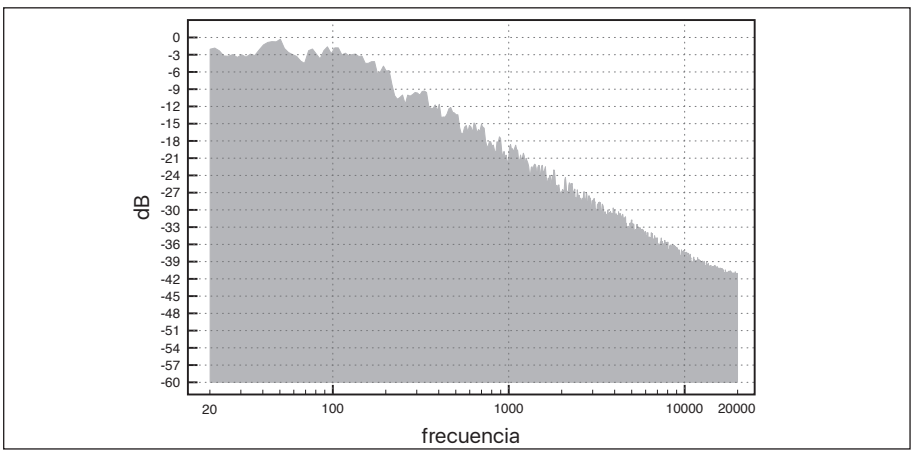


fig. 3.2: espectro del ruido rosado

[2] Otro modo para definir la diferencia entre ruido blanco y ruido rosa es el siguiente: mientras el espectro del ruido blanco tiene la misma energía por cada frecuencia, el espectro del ruido rosa tiene *la misma energía por cada octava*. Dado que una octava del registro agudo ocupa un espacio de frecuencia dos veces mayor que el de la octava anterior, la misma energía total es distribuida en un espacio siempre mayor, y esto determina la atenuación de 3 dB, característica del ruido rosa.

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

En los sistemas digitales, el ruido blanco generalmente es producido mediante generadores de números *aleatorios*: la onda aleatoria resultante contiene todas las frecuencias reproducibles por el sistema digital usado. En realidad, los generadores de números aleatorios utilizan procedimientos matemáticos que no son precisamente aleatorios: en efecto, generan series que se repiten después de un determinado número de eventos. Por eso estos generadores son definidos **generadores pseudoaleatorios**.
Modificando algunos parámetros se pueden generar distintos tipos de ruido. El generador de ruido blanco, por ejemplo, genera muestras aleatorias a la frecuencia de muestreo (es decir, si la frecuencia de muestreo del sistema es de 48000 Hz, son generados 48000 valores aleatorios por segundo); sin embargo, es posible modificar la frecuencia con la cual estos números son generados: utilizando una frecuencia de generación de números igual a 5000 por segundo, por ejemplo, ya no se obtendrá ruido blanco, sino ruido con una fuerte atenuación en las frecuencias agudas.

Cuando la frecuencia de generación de muestras es menor que la frecuencia de muestreo, se crea el problema de cómo "llenar los espacios" entre una muestra y la siguiente: un sistema DSP (ver el glosario, cap. 1T), en efecto, siempre debe producir un número de muestras por segundo igual a la frecuencia de muestreo. Hay varias maneras de resolver este problema; veamos estas tres soluciones:

- **Generadores de muestras pseudoaleatorias simples**.
  Generan valores aleatorios a una frecuencia determinada y mantienen el valor de cada muestra hasta la generación del siguiente valor, creando una señal de escalones. En la fig. 3.3, vemos el gráfico relativo a un generador de ruido a 100 Hz: cada valor aleatorio generado se repite en las siguientes muestras por un periodo igual a 1/100 de segundo, cada vez que se genera un nuevo valor. Si la frecuencia de muestreo fuera de 48000 Hz, por ejemplo, cada valor sería repetido por: 48000/100 = 480 muestras.

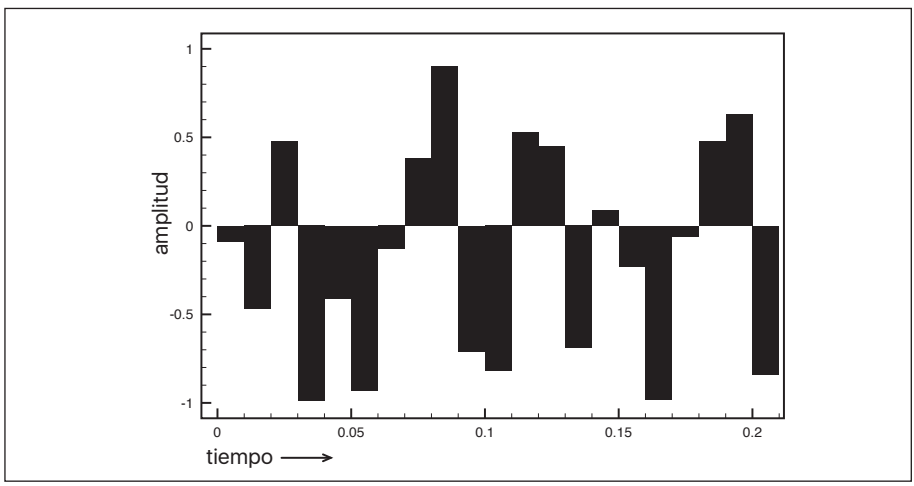


fig. 3.3: generación de valores pseudoaleatorios

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

- **Generadores de muestras pseudoaleatorias con interpolación** entre un número aleatorio y el siguiente (ver la sección de interpolación lineal, en el cap. 2.1): como podemos ver en la fig. 3.4, las muestras que se encuentran entre los valores aleatorios producidos por el generador forman un segmento de recta que lleva gradualmente de un valor a otro.

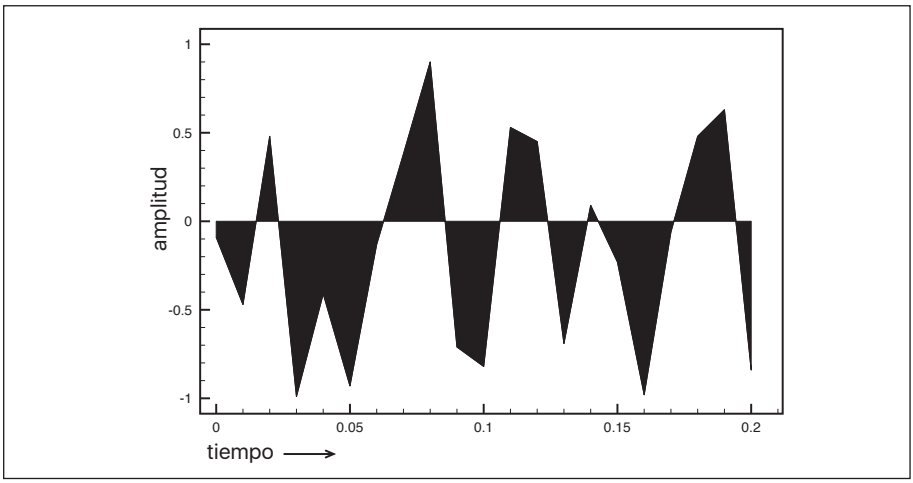


fig. 3.4: generación de valores pseudoaleatorios con interpolación lineal

La interpolación entre un valor y otro puede ser lineal, como la que se ilustra en la figura, o también *polinomial*, es decir, realizada utilizando funciones polinomiales (no profundizaremos en el tema) que unen los valores a través de curvas en lugar de rectas (ver fig. 3.5). Las interpolaciones polinomiales más usadas en la música por computadora son: la interpolación *cuadrática* (realizada con un polinomio de segundo grado) y la *cúbica* (polinomio de tercer grado): normalmente los lenguajes de programación para la síntesis y el procesamiento de señales cuentan con algoritmos específicos para realizar estas interpolaciones.

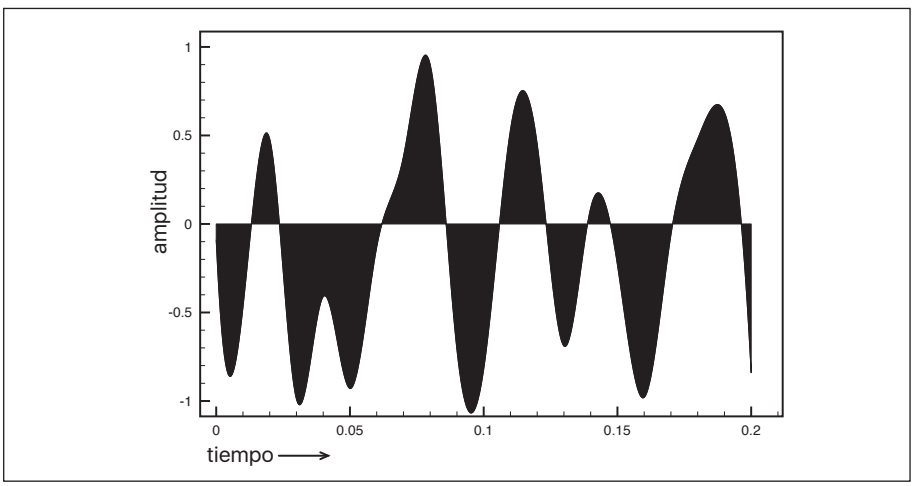


fig. 3.5: generación de valores pseudoaleatorios con interpolación lineal polinomial

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

- **Generadores de muestras pseudoaleatorios con filtro:** en este tipo de sistema, la señal de salida es filtrada mediante un filtro pasa bajos. Hablaremos de este tipo de generador en la sección dedicada a los filtros pasabajos

EJEMPLO INTERACTIVO 3A • *GENERADORES DE RUIDO* - PRESETS 1- 4

OSCILADORES Y OTROS GENERADORES DE SEÑAL

En la sección 1.2 hemos visto algunas formas de onda "clásicas" usadas normalmente en sintetizadores, como la onda cuadrada, la onda diente de sierra y la triangular; en la sección 2.1, además, hemos visto que tales formas de onda, cuando son perfectas geométricamente (o sea, perfectamente cuadradas, triangulares, etc.) contienen un número infinito de componentes. La presencia de infinitas componentes, sin embargo, causa algunos problemas importantes en la reproducción del sonido digital: esto se debe al hecho de que una tarjeta de sonido no puede reproducir frecuencias superiores a la mitad de la frecuencia de muestreo[3] (profundizaremos en este tema en el capítulo 5).
Cuando se intenta reproducir digitalmente un sonido en el cual la frecuencia de las componentes supera la capacidad de la tarjeta de sonido, se obtienen componentes indeseadas, casi siempre inarmónicas. Para evitar este problema, en la música digital a menudo se usan **osciladores de banda limitada**. Tales osciladores reproducen las formas de onda clásicas y son construidos de manera que las componentes nunca superen la mitad de la frecuencia de muestreo. Los sonidos generados por este tipo de osciladores pueden ser, por lo tanto, un buen punto de inicio para obtener sonoridades particulares mediante el uso de filtros, de hecho, son usados ampliamente en la implementación de sintetizadores de síntesis sustractiva. En la sección 3.5 analizaremos la estructura de un sintetizador sustractivo típico.
Naturalmente, en la síntesis sustractiva también será posible utilizar sonidos sintéticos, ricos de parciales realizadas con diversas técnicas (por ejemplo las técnicas de síntesis no lineal o la síntesis por modelos físicos), de las cuales hablaremos en los próximos capítulos.

FILTRADO DE SONIDOS MUESTREADOS

Uno de los usos más comunes de los filtros y de los ecualizadores, que va más allá de la síntesis sustractiva, es el filtrado de sonidos muestreados. A diferencia del ruido blanco, que contiene todas las frecuencias a un mismo valor de amplitud, un sonido muestreado contendrá un número limitado de frecuencias y de relaciones

[3] Es por esto que la frecuencia de una tarjeta de sonido casi siempre es superior al doble de la máxima frecuencia audible al ser humano.

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

de amplitud entre las componentes, que pueden variar dependiendo del sonido en cuestión.
Por eso, antes de efectuar un filtrado, hay que ser conscientes del rango de frecuencias del sonido que se va a procesar. De hecho, como ya se ha dicho antes, *es posible atenuar o hacer resaltar solo las frecuencias que estén contenidas en el sonido original*. Esto también es válido para el procesamiento de sonidos procedentes de una fuente en vivo.

...

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

El capítulo continúa con:

3.2 FILTROS PASA BAJOS, PASA ALTOS, PASA BANDA Y RECHAZA BANDA
Filtro pasa bajos
Filtro pasa altos
Filtro pasa banda
Filtro rechaza banda

3.3 EL FACTOR Q O FACTOR DE RESONANCIA

3.4 ÓRDENES DE FILTROS Y CONEXIÓN EN SERIE
Filtros de primer orden
Filtros de segundo orden
Filtros resonantes de segundo orden
Filtros de orden superior: la conexión en serie

3.5 LA SÍNTESIS SUSTRACTIVA
Anatomía de un sintetizador de síntesis sustractiva

3.6 ECUACIONES DE FILTROS DIGITALES
Filtros no recursivos o filtros FIR
Filtros recursivos o filtros IIR

3.7 FILTROS CONECTADOS EN PARALELO, Y ECUALIZACIÓN GRÁFICA
Ecualizador gráfico

3.8 OTRAS APLICACIONES DE CONEXIÓN EN SERIE: ECUALIZADORES PARAMÉTRICOS Y FILTROS SHELVING
Filtros shelving
Ecualizador paramétrico

3.9 OTRAS FUENTES PARA SÍNTESIS SUSTRACTIVA: IMPULSOS Y CUERPOS RESONANTES
Análisis del comportamiento de un filtro: respuesta al impulso y respuesta en frecuencia

ACTIVIDADES
- Ejemplos interactivos

EVALUACIÓN
- Prueba de preguntas de respuestas cortas
- Prueba de escucha y análisis

MATERIALES DE APOYO
- Conceptos fundamentales - Glosario - Discografía

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

3P
GENERADORES DE RUIDO, FILTROS Y SÍNTESIS SUSTRACTIVA

3.1 FUENTES SONORAS PARA LA SÍNTESIS SUSTRACTIVA
3.2 FILTROS PASA BAJOS, PASA ALTOS, PASA BANDA Y RECHAZA BANDA
3.3 EL FACTOR Q O FACTOR DE RESONANCIA
3.4 ORDEN DE FILTRO Y CONEXIÓN EN SERIE
3.5 SÍNTESIS SUSTRACTIVA
3.6 ECUACIONES DE FILTROS DIGITALES
3.7 FILTROS CONECTADOS EN PARALELO, Y ECUALIZACIÓN GRÁFICA
3.8 OTRAS APLICACIONES DE CONEXIÓN EN SERIE: ECUALIZADORES PARAMÉTRICOS Y FILTROS SHELVING
3.9 OTRAS FUENTES PARA SÍNTESIS SUSTRACTIVA: IMPULSOS Y CUERPOS RESONANTES

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri

© Contemponet 2020

AGENDA DE APRENDIZAJE

PRERREQUISITOS DEL CAPÍTULO

- Contenidos de los Capítulos 1 y 2 (Teoría y Práctica), Capítulo 3 (Teoría), Interludio A

OBJETIVOS

Habilidades

- Saber generar y controlar diferentes tipos de señales complejas para síntesis sustractiva (ruido blanco, ruido rosado, impulsos, etc.)
- Saber construir algoritmos usando filtros pasa bajos, pasa altos, pasa banda, rechaza banda, shelving y resonantes, y aprender a controlarlos usando varios parámetros, Q y orden de filtro
- Saber implementar filtros FIR (no recursivos) y filtros IIR (recursivos)
- Saber construir un sintetizador sustractivo simple
- Saber escribir algoritmos usando filtros conectados en serie y en paralelo
- Saber construir ecualizadores gráficos y paramétricos

Competencias

- Ser capaz de realizar un estudio sonoro corto basado en las técnicas de síntesis sustractiva, y grabarlo en un archivo de audio

CONTENIDOS

- Fuentes para síntesis sustractiva
- Filtros pasa altos, pasa bajos, pasa banda, rechaza banda, shelving y resonantes
- El factor Q y el orden de los filtro
- Filtros FIR y IIR
- Conexión de filtros en serie y en paralelo
- Ecualizadores gráficos y paramétricos

TIEMPO DE DEDICACIÓN - Cap. 3 (Teoría y Práctica) + Interludio B

Autodidactas

Sobre un total de 300 horas de estudio individual (volumen I, teoría y práctica):

- Aproximadamente 110 ore

Cursos

Sobre un total de 60 horas de clase + 120 de estudio individual (volumen I, teoría y práctica):

- Aproximadamente 18 horas de clase frontal + 4 de "feedback"
- Aproximadamente 44 horas de estudio individual

ACTIVIDADES

- Actividades en la computadora: sustitución de partes de algoritmos, corrección, completión y análisis de algoritmos, construcción de nuevos algoritmos.

EVALUACIÓN

- Composición de un estudio sonoro breve - Proyecto integrado de Ingeniería en Reversa

MATERIALES DE APOYO

- Lista de objetos de Max - Lista de atributos de objetos de Max específicos

© Contemponet 2020

3.1 FUENTES SONORAS PARA LA SÍNTESIS SUSTRACTIVA

Como aprendiste en la sección 3.1 de la teoría, el propósito de un filtro es modificar de alguna manera el espectro de una señal. En esta sección, antes de introducir el tema del filtrado en Max, hablaremos de un objeto que puedes usar para mostrar un espectro: el objeto **`spectroscope~`**. Este objeto gráfico se encuentra en la categoría "Audio" de la paleta "Add", (fig. 3.1). Ya sabes que, en alternativa, puedes crear una *object box* y escribir en su interior el nombre "spectroscope~".

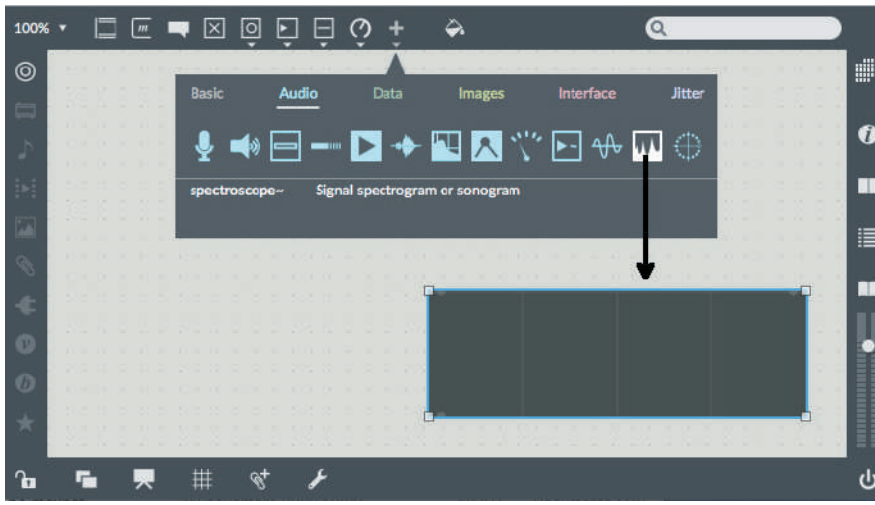


fig. 3.1: objeto `spectroscope~`

Abre el archivo **03_01_spectroscope.maxpat** (fig. 3.2).

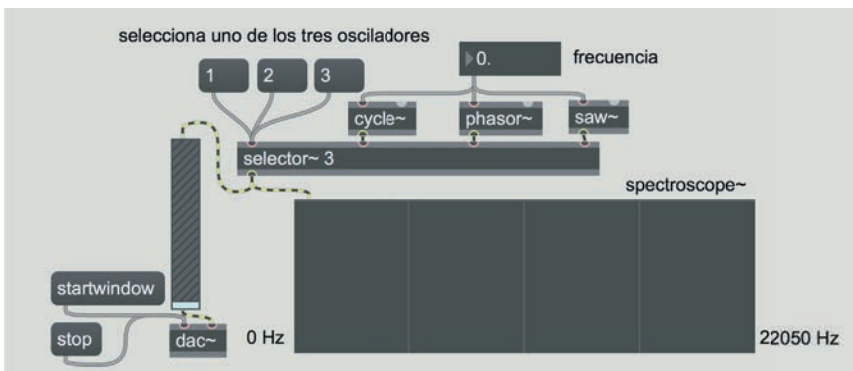


fig. 3.2: archivo **03_01_spectroscope.maxpat**

En este *patch* hemos conectado un objeto `selector~` al espectroscopio para poder alternar entre tres osciladores, uno que genera una onda sinusoidal (`cycle~`), otro que genera una onda diente de sierra no limitada en frecuencia

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

(**phasor~**), y otro más que genera una onda diente de sierra limitada en frecuencia[1] (**saw~**). Las tres *message box* conectadas a la entrada izquierda del objeto **selector~** son usadas para escoger uno de los tres osciladores; para comparar sus espectros debes definir la frecuencia en la *float number box* y luego alternar entre los tres osciladores. Observa que la onda sinusoidal tiene solo una componente, mientras que **phasor~**, siendo un objeto no limitado en frecuencia, tiene un espectro mucho más rico[2].
Asegúrate de probar distintas frecuencias para todas las formas de onda mientras observas la imagen producida por el espectroscopio.
Como ya dijimos, el espectro mostrado es el del sonido entrante: las componentes del sonido son distribuidas de izquierda a derecha a través de una banda de frecuencia que, por defecto, va de 0 a 22050 Hz.
Estos dos valores, las frecuencias mínima y máxima que el espectroscopio puede mostrar, pueden ser modificadas en el *inspector* en la categoría "Value" (valor), usando el atributo "Lo and Hi Domain Display Value" (valores mínimo y máximo a mostrar en el dominio).

Añade un objeto **spectroscope~** a un *patch* que ya hayas creado para que te familiarices con la relación entre el sonido y su contenido espectral. Puedes añadir el objeto a los patches de los capítulos anteriores. En el archivo 01_14_audiofile.maxpat, por ejemplo, puedes tratar de conectar el espectroscopio a la salida izquierda del objeto **sfplay~**, o en el archivo IA_06_random_walk.maxpat, puedes conectarlo a la salida de [**p** monosynth]. ¿Logras ver la relación entre la frecuencia del sonido y la forma del espectro en este último *patch*? (Prueba el preset número 5).

Veamos ahora algunas formas de producir distintos tipos de ruido en Max. El primero que veremos es el ruido blanco, generado por el objeto **noise~** (fig. 3.3).

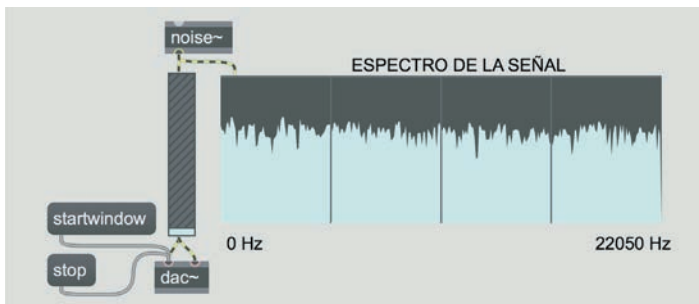


fig. 3.3: generador de ruido blanco

[1] Ver sección 1.2.

[2] El objeto **spectroscope~** usa un algoritmo de análisis espectral llamado Transformada Rápida de Fourier (Fast Fourier Transform). Ya hemos mencionado el teorema de Fourier al final del capítulo 2 de la teoría, y volveremos a hablar en detalle sobre el mismo en otro volumen de la serie.

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

En la figura (que te invitamos a recrear por tu cuenta) hemos conectado el generador de ruido a un objeto **spectroscope~**, a través del cual podemos ver que el espectro del ruido blanco contiene energía en todas las frecuencias. A diferencia de los generadores que hemos visto hasta el momento, el generador de ruido blanco no necesita ningún parámetro: su función es generar una señal a la tasa de muestreo, con valores comprendidos entre -1 y 1 (ver sección 3.1 de la teoría).
El segundo tipo de generador de ruido disponible en Max es el objeto **pink~**, que genera ruido rosado (fig. 3.4).

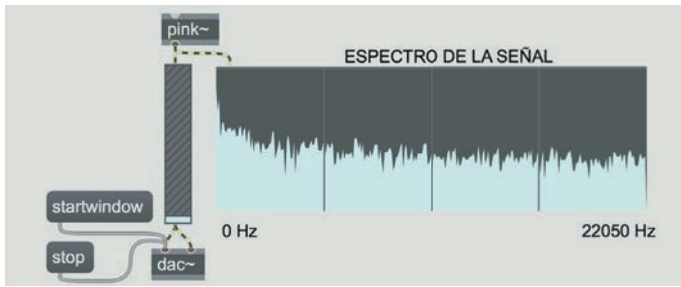


fig. 3.4: ruido rosado

Observa que el espectro del ruido rosado, a diferencia del ruido blanco, presenta una atenuación gradual a medida que la frecuencia aumenta, y esta atenuación es de 3 dB por octava (ver sección 3.1 de la teoría).

Reconstruye el *patch* mostrado en la figura y escucha atentamente la diferencia entre el ruido blanco y el ruido rosado. ¿Cuál de los dos te parece más agradable (o quizás menos desagradable)?¿por qué?

Agrega un osciloscopio (**scope~**) a los dos *patches* que acabas de reconstruir, recordando definir los atributos "Calccount - samples per pixel" (muestras por pixel)[3] en la categoría "Value" del *inspector*, y observa la diferencia entre las formas de onda del ruido blanco y del ruido rosado.
En la figura 3.5 vemos ambas formas de onda lado a lado.

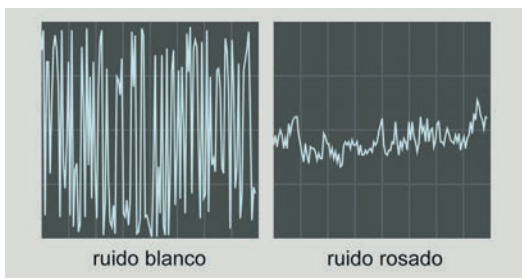


fig. 3.5: formas de onda del ruido blanco y del ruido rosado

[3] Hemos explicado este atributo en la sección 1.2.

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

Sin entrar en detalles técnicos, observa que el ruido blanco es básicamente un torrente de números aleatorios. El ruido rosado, por el contrario, es generado usando un algoritmo más complejo en el que cada muestra, a pesar de ser generada aleatoriamente, no puede diferir mucho del valor de su predecesor. Esto resulta en la forma de onda "serpenteante" que vemos en la figura. El comportamiento de las dos formas de onda demuestra su contenido espectral: cuando la diferencia entre una muestra y la siguiente es grande, la energía de las frecuencias altas en la señal es mayor[4]. Como ya sabes, el ruido blanco tiene más energía en las frecuencias altas que el ruido rosado.

Otro generador interesante es **`rand~`**, que genera muestras aleatorias a una frecuencia que puede ser definida autónomamente, y conecta esos valores usando segmentos de recta (fig. 3.6). A diferencia de `noise~` y `pink~`, que generan muestras aleatorias en cada paso del DSP (produciendo una cantidad de muestras en un segundo igual a la tasa de muestreo), con `rand~` es posible escoger la frecuencia a la que las muestras aleatorias serán generadas; la transición entre un valor de muestra y otro es gradual, gracias a la interpolación lineal.

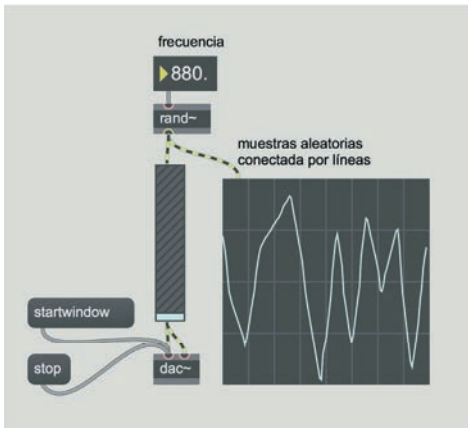


fig. 3.6: objeto `rand~`

Obviamente, este generador produce un espectro que varía de acuerdo a la frecuencia que se ha configurado: muestra una banda primaria de frecuencias entre 0 Hz y la frecuencia configurada, seguido de bandas adicionales que son atenuadas gradualmente y cuyos anchos son también iguales a la frecuencia que se ha configurado. La figura 3.7 muestra este interesante espectro.

4 Para entender esta afirmación, observa que la forma de onda de un sonido agudo oscila rápidamente, mientras que la de un sonido grave oscila lentamente. A amplitudes iguales, en promedio, la diferencia entre muestras consecutivas en el caso de frecuencias altas será mayor que en el caso de frecuencias graves.

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

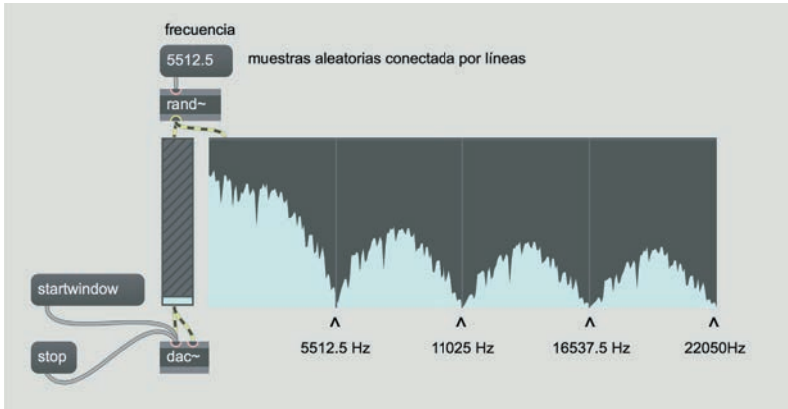


fig. 3.7: espectro generado por el objeto `rand~`

En el ejemplo, podemos ver que la frecuencia del objeto `rand~` es 5512.5 Hz (un cuarto de la máxima frecuencia visible en el espectroscopio de la figura), y la primera banda va entre 0 y 5512.5 Hz. A esta le siguen bandas secundarias, progresivamente atenuadas, todas con un ancho de 5512.5 Hz. Al cambiar la frecuencia de `rand~` se cambia el ancho de las bandas y también su número. Si duplicas la frecuencia a 11025 Hz, por ejemplo, verás dos bandas, precisamente, cada una con un ancho de 11025 Hz.

Otro generador de ruido es **`vs.rand0~`**[5] (el último carácter antes de la tilde es un cero), que al igual `rand~`, genera muestras aleatorias a una frecuencia dada pero no interpola, manteniendo el valor de cada muestra hasta que se genere una nueva, produce cambios escalonados en el valor. El espectro de este objeto está dividido en bandas, al igual que `rand~` en la figura 3.7, pero como puedes ver en la figura 3.8, la atenuación de las bandas secundarias es mucho menor debido a los cambios abruptos entre valores de muestra.

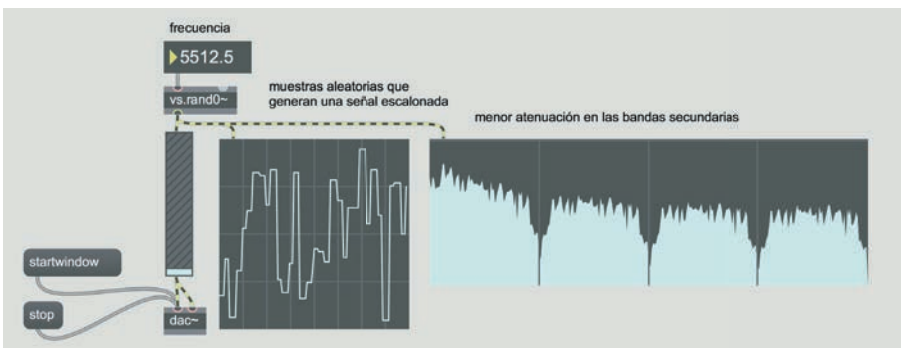


fig. 3.8: objeto `vs.rand0~`

[5] Ya hemos dicho en los capítulos anteriores que el prefijo "vs" al inicio del nombre indica que este objeto es parte de la biblioteca Virtual Sound Macros.

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

En la biblioteca *Virtual Sound Macros* también hemos provisto un generador de ruido que usa interpolación cúbica, llamado **`vs.rand3~`** (fig. 3.9).

fig. 3.9: objeto `vs.rand3~`

Gracias a la interpolación polinomial de este objeto, las transiciones entre una muestra y la siguiente son suaves, como puedes ver en el osciloscopio. Las transiciones forman una curva en vez de una serie de segmentos de línea conectados, y el efecto resultante es una atenuación fuerte de las bandas secundarias. Recrea los *patches* encontrados en las figuras 3.6 a 3.9 para experimentar con distintos generadores de ruido.

Los osciladores "clásicos" (aquellos que producen ondas diente de sierra, cuadradas y triangulares) son otras fuentes de sonido ricas en componentes, lo que los hace efectivos en el uso con filtros. En la sección 1.2 examinamos tres osciladores de banda limitada que generan estas ondas: `saw~`, `rect~`, y `tri~`. Usaremos estos osciladores frecuentemente en el transcurso de este capítulo. En la sección 3.1 de la teoría también hemos aprendido sobre la posibilidad de filtrar sonidos muestreados; en este capítulo también daremos algunos ejemplos de filtrado de este tipo de sonidos usando el objeto **`sfplay~`** (del cual ya hablamos en la sección 1.5).

...

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

El capítulo continúa con:

3.2 FILTROS PASA BAJOS, PASA ALTOS, PASA BANDA Y RECHAZA BANDA

3.3 EL FACTOR Q O FACTOR DE RESONANCIA

3.4 ORDEN DE FILTRO Y CONEXIÓN EN SERIE
El objeto umenu
Filtros de primer orden
Filtros de segundo orden
Filtros de orden mayor: conexiones en serie

3.5 SÍNTESIS SUSTRACTIVA
Comunicaciones "inalámbricas": el objeto pvar
Múltiples alternativas: el objeto radiogroup
Anatomía de un sintetizador sustractivo

3.6 ECUACIONES DE FILTROS DIGITALES
Filtros no recursivos o filtros FIR
Filtros recursivos o filtros IIR

3.7 FILTROS CONECTADOS EN PARALELO, Y ECUALIZACIÓN GRÁFICA
Uso de un banco de filtros paralelo
Ecualizador gráfico

3.8 OTRAS APLICACIONES DE CONEXIÓN EN SERIE: ECUALIZADORES PARAMÉTRICOS Y FILTROS SHELVING
Ecualización paramétrica

3.9 OTRAS FUENTES PARA SÍNTESIS SUSTRACTIVA: IMPULSOS Y CUERPOS RESONANTES

Actividades
• Actividades en la computadora: sustitución de partes de algoritmos, corrección, compleción y análisis de algoritmos, construcción de nuevos algoritmos.

Evaluación
• Composición de un estudio sonoro breve - Proyecto integrado de ingeniería en reversa

Materiales de apoyo
• Lista de objetos de Max - Lista de atributos de objetos de Max específicos

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

Interludio B
OTROS ELEMENTOS DE PROGRAMACIÓN CON MAX

IB.1 NOTAS SOBRE MIDI
IB.2 EL OPERADOR MÓDULO Y LA RECURSIÓN
IB.3 RUTEAR SEÑALES Y MENSAJES
IB.4 LOS OPERADORES RELACIONALES Y EL OBJETO SELECT
IB.5 DESCOMPONER UNA LISTA: EL OBJETO ITER
IB.6 BUCLE DE DATOS
IB.7 GENERAR UNA LISTA ALEATORIA
IB.8 CÁLCULOS Y CONVERSIONES CON MAX
IB.9 USO DE TABLAS PARA ENVOLVENTES: EL TONO SHEPARD

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri

© Contemponet 2020

AGENDA DE APRENDIZAJE

PRERREQUISITOS DEL CAPÍTULO

- Contenidos de los capítulos 1, 2 y 3 (TEORÍA Y PRÁCTICA), INTERLUDIO A

OBJETIVOS

Habilidades

- Saber utilizar diferentes objetos y señales MIDI simples
- Saber utilizar operaciones recursivas y conversiones en Max
- Saber construir un arpegiador, también con el uso de intervalos probabilísticos
- Saber rutear señales y mensajes seleccionando entradas y salidas
- Saber confrontar valores y analizar su relación
- Saber descomponer listas de datos
- Saber construir secuencias repetitivas a través de bucles de datos
- Saber generar listas aleatorias para la simulación de cuerpos resonantes.
- Saber construir un Tono Shepard, o glisando infinito

CONTENIDOS

- Uso básico del protocolo MIDI
- Operaciones recursivas y secuencias repetitivas
- Arpegiadores e intervalos probabilísticos
- Confronto de valores, conversiones y clasificación de señales y mensajes
- Descomposición de listas y generación de listas aleatorias
- Tono Shepard

TIEMPO DE DEDICACIÓN - Capítulo 3 (Teoría y Práctica) +Interludio B

Autodidactas

Sobre un total de 300 horas de estudio individual (volumen I, teoría y práctica):

- Aproximadamente 110 horas

Cursos

Sobre un total de 60 horas de clase + 120 de estudio individual (volumen I, teoría y práctica):

- Aproximadamente 18 horas de clase frontal + 4 de "feedback"
- Aproximadamente 44 horas de estudio individual

ACTIVIDADES

Actividades en el computador

- Sustitución de partes de algoritmos, corrección, compleción y análisis de algoritmos, construcción de nuevos algoritmos

EVALUACIÓN

- Proyecto integrado: ingeniería en reversa

MATERIALES DE APOYO

- Lista de objetos Max - Lista de mensajes, atributos y parámetros de objetos Max específicos - Glosario de términos usados en este capítulo

© Contemponet 2020

IB.1 NOTAS SOBRE MIDI

El MIDI es un sistema de comunicación entre computadoras e instrumentos musicales electrónicos y/o digitales: a través de este sistema es posible, por ejemplo, conectar (con un cable MIDI adecuado) una computadora a un sintetizador para que este último pueda ser "tocado" por la computadora. En otras palabras, gracias al MIDI la computadora es capaz determinar qué notas tocar en el sintetizador, con qué intensidad, con qué duración, etc.
Los instrumentos musicales digitales también pueden ser "virtuales", a saber, aplicaciones instaladas en nuestra computadora que simulan el comportamiento de instrumentos digitales reales. La comunicación con instrumentos virtuales a través de MIDI también es posible, y se logra realizando una conexión de tipo virtual (con un cable no material) entre el programa que envía comandos MIDI (Max, por ejemplo) y el programa (el instrumento virtual) que los recibe. En el capítulo 9 podremos ver en detalle diferentes objetos de Max que usan el protocolo MIDI. Considerando que después de este Interludio serán utilizados algunos objetos que manejan mensajes MIDI, en esta sede les proporcionaremos una breve introducción de estos objetos, dejando los detalles para el capítulo 9 de las secciones de teoría y de práctica.

Abre el archivo **IB_01_notas_MIDI.maxpat**: en la figura IB.1 vemos la parte superior de la Patcher Window.

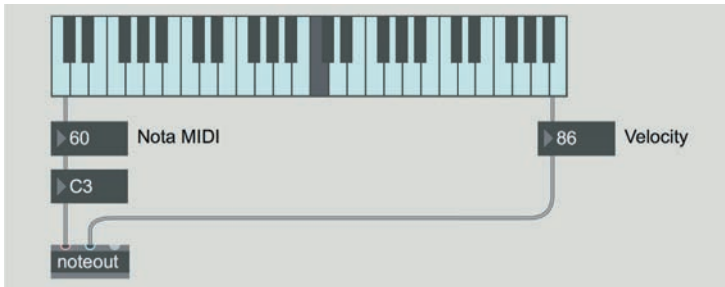


fig. IB.1: archivo **IB_01_notas_MIDI.maxpat**, parte superior

Hemos conectado el objeto `kslider` (el teclado musical) a algunas *number box*, y a través de estas, el objeto **`noteout`**. Como podemos ver, haciendo click en la tecla de `kslider` obtendremos en la salida izquierda el valor MIDI de la nota seleccionada (ver también la sección 1.4) y en la salida derecha el valor de *velocity*, o sea la intensidad de la nota; haciendo click en la parte alta de una tecla de `kslider` se obtienen valores altos de *velocity*, haciendo click en la parte baja se obtienen valores bajos. La velocity puede variar entre 1 y 127. Los valores de nota MIDI y de *velocity* son enviados a las entradas izquierda y central del objeto `noteout`; este último envía al instrumento (real o virtual) al que está conectado[1], el comando relativo a la ejecución de la nota.

[1] La entrada derecha del objeto `noteout` sirve para configurar el canal MIDI, que por el momento no necesitamos. Para más detalles, ver el capítulo 9.

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

En el protocolo MIDI este comando se define como "**note-on**". Si haces click en una tecla de `kslider` (lo suficientemente alta como para obtener una *velocity* alta, superior a 90) se debería escuchar un sonido de piano. Este sonido no es de Max, sino de un instrumento virtual contenido en el sistema operativo de tu computadora que por defecto, está configurado con un sonido de piano. Si tratas de tocar varias notas te darás cuenta de que, por cada pareja nota-*velocity* que el objeto `noteout` recibe, se escucha una nueva nota, pero las anteriores no se interrumpen: es como si las teclas se quedaran "pegadas". Lo que sucede es que a través de `noteout` le estamos ordenando al instrumento virtual que inicie una nota, ¡mas no que la detenga!
Para detener una nota debemos enviar nuevamente el relativo valor MIDI asociado a una *velocity* igual a 0. El valor de *velocity* 0 corresponde al comando "**note-off**", que equivale a decir "levanta el dedo de la tecla".
Para poder "apagar" una nota MIDI usando `kslider`, debemos cambiar la manera como este interpreta los mensajes de nota MIDI: hay que ir al modo edición y seleccionar el *inspector* del `kslider` superior, localizar la categoría "Value" y cambiar en el menú la pestaña correspondiente al parámetro "**Display Mode**" de "**Monophonic**" a "**Polyphonic**", para luego regresar al modo ejecución.
Ahora, la primera vez que se haga click en una tecla de `kslider` escucharemos una nota con la *velocity* correspondiente; un segundo click sobre la misma tecla enviará de nuevo la nota, pero con una *velocity* igual a 0, que hace que finalice el sonido: ¡inténtalo! Este modo se define como "Polyphonic", porque a diferencia del modo "Monophonic", nos permite tener activas varias notas simultáneamente.
En la figura IB.2 vemos la parte inferior del archivo **IB_01_notas_MIDI.maxpat**.

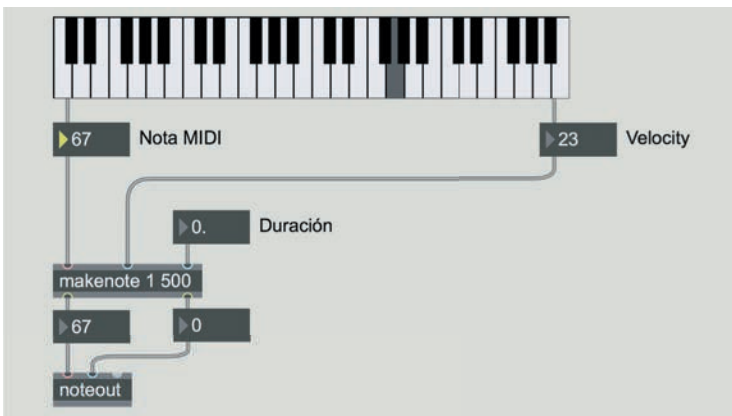


fig. IB.2: archivo **IB_01_notas_MIDI.maxpat**, parte inferior

Hemos conectado el objeto `kslider` (en el modo "monophonic") al objeto **`makenote`**, el cual, cada vez que recibe un comando MIDI *note-on* genera el comando MIDI *note-off* correspondiente después de un intervalo temporal establecido. El objeto tiene tres entradas: una destinada al valor de nota MIDI, otra a la velocity, y otra a la duración en milisegundos (relativa al tiempo que transcurre entre un *note-on* y el sucesivo *note-off*); tiene también dos salidas, una para el valor de nota MIDI y la otra para la *velocity*.

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

Los parámetros son dos, la *velocity* y la duración en milisegundos; en el *patch* estos valores son: 1 para la *velocity* y 500 milisegundos (medio segundo), correspondientes a la duración. Cuando el objeto recibe un *note-on* lo envía directamente a las salidas, después espera la duración establecida (500 milisegundos) y envía el mensaje *note-off*. Observa que la *velocity* que enviamos a través de `kslider` (el valor 103, según la figura IB.2) anula y sustituye el valor 1 que habíamos escrito como argumento: de hecho, este último se ha utilizado solamente para poder escribir el segundo argumento (o sea la duración), que como tal, ¡necesariamente debe estar precedido por el primero!
La duración también puede ser modificada enviando el nuevo valor a la entrada de la derecha, que sustituirá el valor que hemos utilizado como segundo argumento.
Trata de tocar algunas notas y de cambiar la duración usando el objeto `makenote`: observa cómo se genera, de su segunda salida, un valor de *velocity* idéntico al generado por `kslider`, y después del tiempo establecido, el valor 0.
Ahora vamos a añadir un sumador a la parte inferior del *patch* (fig. IB.3).

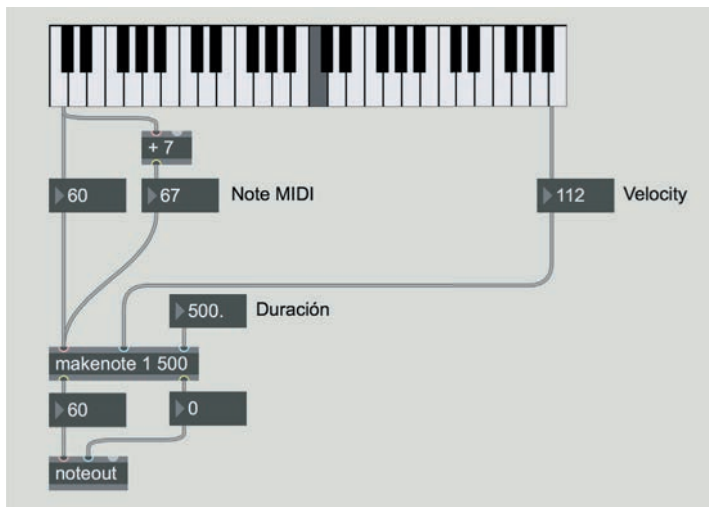


fig. IB.3: transposición MIDI

Este *patch* es similar al del archivo **IA_01_transposición.maxpat**, que hemos estudiado en la primera sección del interludio A. También en este caso, por cada tecla tocada en el objeto `kslider` son generadas dos notas a distancia de 7 semitonos, o sea una quinta. Cada vez que hacemos click en una tecla de `kslider`, el valor de nota MIDI correspondiente es enviado al objeto `makenote` y, simultáneamente, a un sumador que añade a la nota el valor 7, antes de enviarla a `makenote`. Observa que para obtener estas parejas de notas no hemos tenido necesidad de doblar también el valor de la *velocity*, ni el de la duración: estos últimos, en efecto, corresponden a las entradas "frías" del objeto `makenote` y solamente actualizan las variables internas del objeto. El contenido de estas variables internas es reutilizado cada vez que un nuevo valor llega a la entrada "caliente": en el caso de la figura, por ejemplo, las dos notas (DO y SOL centrales) tienen una *velocity* igual a 112 y una duración de 500 milisegundos.

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

En este *patch* podemos ver que la regla según la cual, la entrada "caliente" de un objeto es la que está más a la izquierda, es el lógico complemento de la regla del orden de ejecución de derecha a izquierda (ver la sección 1.7): los primeros mensajes que deben ser transmitidos son los que están más a la derecha, que se dirigen a entradas "frías" y actualizan las variables internas de un objeto (por ejemplo la *velocity* en `makenote`), mientras que los últimos son los que están más a la izquierda, que se dirigen a entradas "calientes" y provocan un output después de que todas las variables internas han sido actualizadas.

...

El capítulo continúa con:

IB.2 EL OPERADOR MÓDULO Y LA RECURSIÓN
La recursión
Construyamos un arpegiador

IB.3 RUTEAR SEÑALES Y MENSAJES

IB.4 LOS OPERADORES RELACIONALES Y EL OBJETO SELECT
El objeto select
Un metrónomo "probabilístico"

IB.5 DESCOMPONER UNA LISTA: EL OBJETO ITER

IB.6 BUCLE DE DATOS

IB.7 GENERAR UNA LISTA ALEATORIA

IB.8 CÁLCULOS Y CONVERSIONES CON MAX
El objeto expr
Convertir intervalos de valores en señales

IB.9 USO DE TABLAS PARA ENVOLVENTES: EL TONO SHEPARD
Uso de tablas de ondas para generar envolventes consecutivas

Actividades
• Sustitución de partes de algoritmos, corrección, compleción y análisis de algoritmos, construcción de nuevos algoritmos

Evaluación
• Proyecto integrado: ingeniería en reversa

Materiales de apoyo
• Lista de objetos Max - Lista de mensajes, atributos y parámetros de objetos Max específicos - Glosario

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

4T
SEÑALES DE CONTROL

4.1 SEÑALES DE CONTROL: EL PANEO ESTÉREO
4.2 DC OFFSET
4.3 SEÑALES DE CONTROL PARA FRECUENCIA
4.4 SEÑALES DE CONTROL PARA AMPLITUD
4.5 MODULACIÓN DEL CICLO DE TRABAJO (MODULACIÓN POR ANCHO DE PULSOS)
4.6 SEÑALES DE CONTROL PARA FILTROS
4.7 OTROS GENERADORES DE SEÑALES DE CONTROL
4.8 SEÑALES DE CONTROL: EL PANEO MULTICANAL

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

AGENDA DE APRENDIZAJE

PRERREQUISITOS DEL CAPÍTULO

- Contenido de los capítulos 1, 2, 3 (teoría)

OBJETIVOS

Conocimientos

- Conocer la teoría y el uso de los parámetros de los osciladores de baja frecuencia (LFO)
- Conocer el uso del DC Offset aplicado a los LFO
- Conocer el uso de la modulación de frecuencia para el vibrato
- Conocer el uso de la modulación de amplitud para el trémolo
- Conocer el uso de la modulación por ancho de pulsos (Pulse Width Modulation)
- Conocer el uso de los LFO para generar señales de control para filtros
- Conocer el uso de generadores de señales pseudo-aleatorias como LFO de control
- Conocer el uso de los osciladores de control para el desplazamiento de sonido en los sistemas estereofónicos y multicanal.

Habilidades

- Saber identificar auditivamente las modificaciones de los parámetros fundamentales de un LFO y saberlos describir

CONTENIDOS

- Osciladores de baja frecuencia: depth, rate e delay
- Manejo de parámetros de LFO y uso de DC Offset
- Manejo de vibrato, trémolo y PWM mediante LFO
- Manejo de los parámetros de filtros mediante LFO
- Desplazamiento del sonido en sistemas estéreo y multicanal
- Osciladores de control modulados por otros LFO

TIEMPO DE DEDICACIÓN - Capítulo 4 (Teoría y Práctica)

Autodidactas

Sobre un total de 300 horas de estudio individual (volumen I, teoría y práctica):

- Aproximadamente 30 horas

Cursos

Sobre un total de 60 horas de clase + 120 de estudio individual (volumen I, teoría y práctica):

- Aproximadamente 5 horas de clase frontal + 1 de "feedback"
- Aproximadamente 12 horas de estudio individual

ACTIVIDADES

- Ejemplos interactivos

EVALUACIÓN

- Prueba de preguntas de respuestas cortas
- Prueba de escucha y análisis

MATERIALES DE APOYO

- Conceptos fundamentales - Glosario

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

4.1 SEÑALES DE CONTROL: EL PANEO ESTÉREO

Tal como vimos en el capítulo 1, es posible variar los parámetros de un sonido (por ejemplo la frecuencia o la amplitud) a través de envolventes que describan en el tiempo el comportamiento de los parámetros en cuestión. Las señales que no sirven para generar un sonido sino para variar sus características (como las que controlan las envolventes) se llaman **señales de control**. Hasta ahora hemos usado solamente segmentos de recta o de exponencial como señales de control. Esta técnica es eficaz cuando tenemos que describir el cambio de un parámetro a través de pocos valores: por ejemplo, para realizar una envolvente ADSR necesitamos 4 segmentos, y el glisando de una nota puede ser realizado con una sola curva exponencial. Sin embargo, los parámetros de un sonido pueden variar también de manera más compleja. Pensemos por ejemplo en el vibrato de un instrumento de cuerdas: este efecto no es más que una oscilación continua de la frecuencia de la nota alrededor de una altura central. Para simular esta vibración necesitaríamos decenas o centenas de segmentos, pero evidentemente, sería poco práctico y muy laborioso. Por el contrario, podemos utilizar un **oscilador de control**, esto es, un oscilador que no sirve para producir sonidos sino simplemente valores que varían de un mínimo a un máximo a una determinada velocidad y que generalmente son asignados a los parámetros de los *osciladores audio*. Tales valores pueden ser asignados también a los parámetros de otros algoritmos de síntesis y de procesamiento de señales.
Es importante notar que los osciladores de control son *osciladores de baja frecuencia* (**LFO, Low Frequency Oscillators**): lo que significa que oscilan a frecuencias que generalmente son inferiores a 30 Hz, y producen valores de control en continuo movimiento, que siguen la forma de onda propia del oscilador. Cada valor de amplitud instantánea de la onda generada por el oscilador de control corresponde a un valor numérico que luego es aplicado a los parámetros audio que queramos.
Hagamos un ejemplo: en la figura 4.1 vemos un LFO que controla la posición de sonido en el espacio y genera una sinusoide que oscila entre MIN (valor mínimo) y MAX (valor máximo).

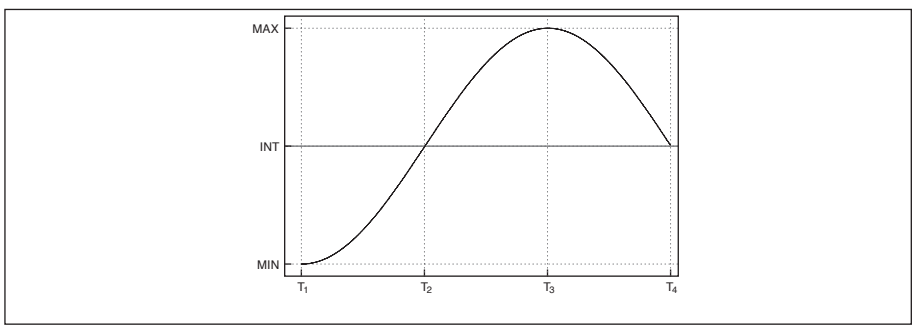


fig. 4.1: LFO que controla la posición de sonido en el espacio

Los valores mínimo y máximo se refieren a los valores de *amplitud* del oscilador de control (usualmente definida como **depth**, que en español significa profundidad), la velocidad de la oscilación, en cambio, depende de la *frecuencia* del oscilador de control (usualmente definida como **rate**, que en español significa frecuencia).

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

Los valores de amplitud instantánea de la sinusoide generada por un LFO (u oscilador de control) son usados como multiplicadores de los valores de amplitud de los dos canales en salida de un oscilador audio: cuando la sinusoide del oscilador de control alcance el valor mínimo (MIN), el sonido del oscilador audio estará completamente a la izquierda; cuando el oscilador de control alcance el valor máximo (MAX), el sonido del oscilador audio estará completamente a la derecha; cuando la sinusoide del oscilador de control se encuentre en correspondencia del valor intermedio (INT), los valores de amplitud del sonido en los canales izquierdo y derecho serán iguales, con lo cual el sonido será percibido al centro.
Obviamente, también es posible utilizar otras formas de onda (triangular, aleatoria, etc.) para variar estos parámetros de control; por ejemplo, si usamos una onda cuadrada, el desplazamiento de derecha a izquierda y viceversa no ocurrirá de manera continua, como con la sinusoide, sino de manera alternada (MIN-MAX-MIN-MAX etc.).

EJEMPLO INTERACTIVO 4A • *Paneo mediante LFO con diferentes formas de onda*

La velocidad (o sea el *rate*) con la cual tales valores oscilan depende de la frecuencia que le asignemos al oscilador de control. Si utilizamos la frecuencia 1, tendremos una oscilación entre MAX y MIN que completa una oscilación una vez por segundo; si aplicamos una frecuencia igual a 0.2 tendremos una oscilación completa cada 5 segundos. ¿Y si utilizamos una frecuencia de 220? La oscilación sería demasiado rápida para poder percibir el desplazamiento de derecha a izquierda y de vuelta (220 veces por segundo): además, esta frecuencia entraría en el campo auditivo y esto añadiría nuevas componentes al espectro del sonido resultante, como veremos en el capítulo 10, en la sección dedicada a la modulación de amplitud.

EJEMPLO INTERACTIVO 4B • *Paneo mediante LFO sinusoidal a diferentes frecuencias*

Con los osciladores de control es posible controlar, de forma similar al procedimiento descrito antes, la amplitud (*depth*) y la frecuencia (*rate*) de un vibrato, de un trémolo, y de la variación de los parámetros de un filtro, tal como veremos en las próximas secciones.

...

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

El capítulo continúa con:

4.2 DC OFFSET

4.3 SEÑALES DE CONTROL PARA FRECUENCIA
El vibrato
Depth del vibrato
Rate del vibrato

4.4 SEÑALES DE CONTROL PARA AMPLITUD

4.5 MODULACIÓN DEL CICLO DE TRABAJO (MODULACIÓN POR ANCHO DE PULSOS)

4.6 SEÑALES DE CONTROL PARA FILTROS

4.7 OTROS GENERADORES DE SEÑALES DE CONTROL
Controlar un sintetizador sustractivo con un lfo

4.8 SEÑALES DE CONTROL: EL PANEO MULTICANAL

Actividades
- Ejemplos interactivos

Evaluación
- Prueba de preguntas de respuestas cortas
- Prueba de escucha y análisis

Materiales de apoyo
- Conceptos fundamentales - Glosario

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

4P
SEÑALES DE CONTROL

4.1 SEÑALES DE CONTROL: PANEO ESTÉREO
4.2 DC OFFSET
4.3 SEÑALES DE CONTROL PARA FRECUENCIA
4.4 SEÑALES DE CONTROL PARA AMPLITUD
4.5 MODULACIÓN DEL CICLO DE TRABAJO (MODULACIÓN POR ANCHO DE PULSOS)
4.6 SEÑALES DE CONTROL PARA FILTROS
4.7 OTROS GENERADORES DE SEÑALES DE CONTROL
4.8 SEÑALES DE CONTROL: PANEO MULTICANAL

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

AGENDA DE APRENDIZAJE

PRERREQUISITOS DEL CAPÍTULO

- Contenidos de los Capítulos 1, 2 y 3 (Teoría y Práctica), Capítulo 4 (Teoría), Interludios A y B

OBJETIVOS

Habilidades

- Saber desplazar un sonido dentro del campo estéreo
- Saber implementar efectos de vibrato
- Saber simular instrumentos cuya frecuencia es controlada, como el Theremin
- Saber implementar efectos de trémolo
- Saber construir algoritmos de modulación de ancho de pulso
- Saber variar la frecuencia de corte, la frecuencia central y el Q de filtros usando señales de control oscilantes
- Saber usar generadores de señales pseudo-aleatorias para control
- Saber desplazar sonidos en un sistema de 4 o más canales usando señales de control

Competencias

- Ser capaz de realizar un estudio sonoro corto basado en la técnica de control de parámetros usando LFOs

CONTENIDOS

- Osciladores de baja frecuencia: depth, rate y delay
- Manejo de parámetros de LFO y uso de DC offset
- Manejo de vibrato, de trémolo y modulación de ancho de pulso mediante LFO
- Manejo de los parámetros de un filtro mediante LFO
- Señales de control pseudo aleatorias
- Desplazamiento de sonido en estéreo y en sistemas multicanal

TIEMPO DE DEDICACIÓN - Capítulo 4 (Teoría y Práctica)

Autodidactas

Sobre un total de 300 horas de estudio individual (volumen I, teoría y práctica):

- Aproximadamente 30 horas

Cursos

Sobre un total de 60 horas de clase + 120 de estudio individual (volumen I, teoría y práctica):

- Aproximadamente 5 horas de clase frontal + 1 de "feedback"
- Aproximadamente 12 horas de estudio individual

ACTIVIDADES

- Actividades en la computadora: sustitución de partes de algoritmos, corrección, completión y análisis de algoritmos, construcción de nuevos algoritmos.

EVALUACIÓN

- Composición de un estudio sonoro breve - Proyecto integrado de Ingeniería en Reversa

MATERIALES DE APOYO

- Lista de objetos de Max - Atributos para objetos específicos de Max - Glosario

© Contemponet 2020

4.1 SEÑALES DE CONTROL: PANEO ESTÉREO

Para posicionar una señal dentro de un campo estéreo, tal como se describe en la sección 4.1 de la teoría, podemos usar la salida de un objeto `cycle~` normal como señal de control de onda sinusoidal. La frecuencia del objeto `cycle~`, en este caso, debe ser lo suficientemente baja como para estar bajo el umbral de escucha humano.

En la sección 1.6 aprendiste cómo definir la posición estéreo de una señal[1]; toma como punto de referencia el archivo **01_18_pan_function.maxpat** para reconstruir en un nuevo *patch* el algoritmo para el posicionamiento de un sonido en un campo estéreo. Usa las partes del *patch* original que estaban conectadas al objeto `line~` (fig. 4.1).

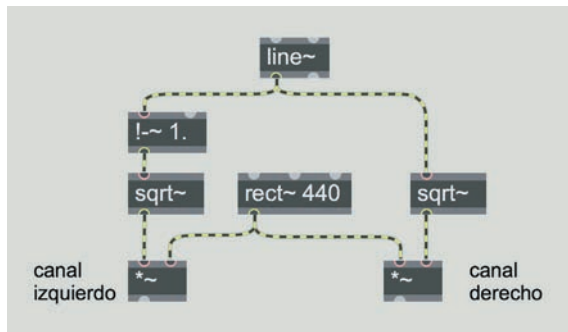


fig. 4.1: algoritmo de paneo

Tienes que reemplazar el objeto `line~` del *patch* original (que modula la posición del sonido en el espacio usando segmentos de línea) por un generador de onda sinusoidal que oscila entre 0 y 1 (un valor de 0 posiciona el sonido a la izquierda, mientras que un valor de 1 lo posiciona a la derecha). El objeto `cycle~`, sin embargo, genera una onda sinusoidal que oscila entre -1 y 1. Puedes modificar el intervalo de oscilación usando algunos cálculos simples que aprendiste en el capítulo teórico, pero eso será tema de la siguiente sección.

[1] Si no recuerdas cómo hacer esto, refresca tu memoria volviendo a leer las secciones relevantes tanto teóricas como prácticas.

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

Por ahora, usa el objeto **scale~** (del cual hablamos en la sección IB.8) para reescalar la señal, y completa el *patch* como se muestra en la figura 4.2.

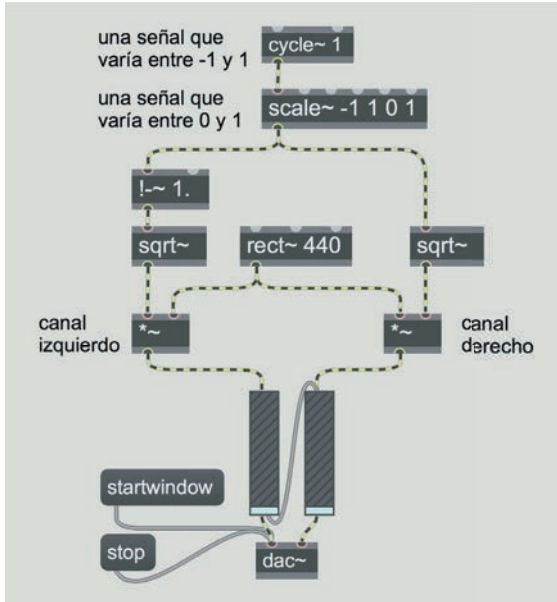


fig. 4.2: paneo estéreo controlado por un LFO

En este *patch*, el objeto **line~** ha sido reemplazado por el objeto **scale~**, y un objeto **cycle~** ha sido conectado a él. Como sabes, el objeto **scale~** tiene cuatro argumentos: los dos primeros especifican el rango de la señal entrante, y los dos últimos especifican el rango deseado para la señal saliente. En nuestro caso, los argumentos [-1 1 0 1] indican que estaremos alimentando el objeto con una señal de entrada que va entre -1 y 1, y que queremos reescalar esta entrada a una señal con rango entre 0 y 1. El objeto **cycle~** en sí mismo está configurado para generar una señal de control de 1 Hz, que hará que el sonido viaje del parlante izquierdo al derecho y viceversa en un periodo de 1 segundo; conectando una *float number point* al objeto **cycle~** puedes cambiar la frecuencia de oscilación.
Prueba este *patch* con varias frecuencias, pero no superes los 20 Hz, puesto que frecuencias más altas generan fenómenos de modulación que trataremos en el capítulo 10.
Podemos simplificar el *patch* usando el objeto multicanale **mc.stereo~**, que es parte de la biblioteca *Virtual Sound Macros*; este objeto implementa un algoritmo de paneo estéreo tomando el sonido posicionado en su entrada izquierda, y colocándolo en el campo estéreo según la señal de control que reciba en su entrada derecha (fig. 4.3).

Puedes ver que el objeto **mc.stereo~** realiza la misma función que el algoritmo de la figura 4.1. En realidad, nos permite "liberar espacio" en la parte gráfica de nuestro *patch*, pero además, es más eficiente. Observa que hemos sustituido los dos **gain~** y el objeto **dac~** con un **mc.gain~** y un **mc.dac~**.

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

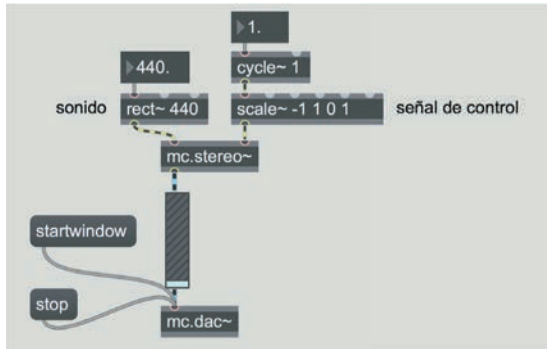


fig. 4.3: paneo estéreo usando el objeto `mc.stereo~`

Es posible usar como señal de control otras formas de onda, por ejemplo la onda cuadrada (fig.4.4).

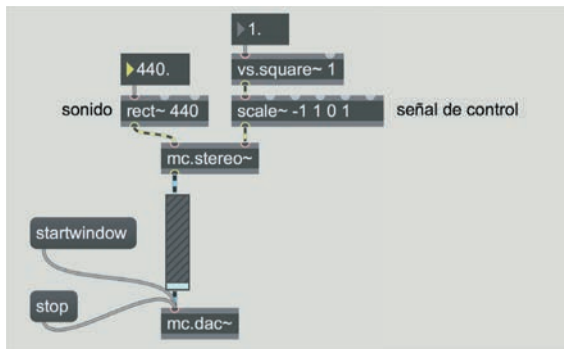


fig. 4.4: control de paneo con un LFO de onda cuadrada

Bajo el control de una onda cuadrada el sonido se mueve de canal en canal sin pasar por posiciones intermedias. Esta discontinuidad, sin embargo, genera un click no deseado en la señal de salida, que puede ser eliminado filtrando la señal de control con un filtro pasa bajos; este filtro suaviza las esquinas de la señal cuadrada (fig. 4.5).

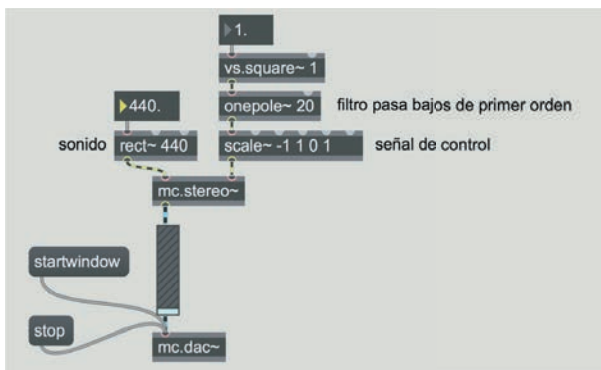


fig. 4.5: filtrado de un LFO

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

En este *patch* tenemos una frecuencia de corte de 20 Hz, lo que significa que la señal de control no puede "saltar" de un valor a otro valor en un tiempo inferior a 1/20 de segundo. Intenta cambiar la frecuencia de corte del filtro para entender mejor cómo influye en la trayectoria del sonido: a menor frecuencia de corte, más suaves serán las transiciones entre canales.

...

El capítulo continúa con:

4.2 DC OFFSET

4.3 SEÑALES DE CONTROL PARA FRECUENCIA
Simulación de un theremin

4.4 SEÑALES DE CONTROL PARA AMPLITUD

4.5 MODULACIÓN DEL CICLO DE TRABAJO (MODULACIÓN POR ANCHO DE PULSOS)

4.6 SEÑALES DE CONTROL PARA FILTROS

4.7 OTROS GENERADORES DE SEÑALES DE CONTROL
La matriz de modulación

4.8 SEÑALES DE CONTROL: PANEO MULTICANAL

Actividades
• Actividades en la computadora: sustitución de partes de algoritmos, corrección, compleción y análisis de algoritmos, construcción de nuevos algoritmos.

Evaluación
• Composición de un estudio sonoro breve - Proyecto integrado de ingeniería en reversa

Materiales de apoyo
• Lista de objetos de Max - Atributos para objetos específicos de Max - glosario

extracto del libro "Música Electrónica y Diseño Sonoro" - vol. 1 - Alessandro Cipriani y Maurizio Giri
© Contemponet 2020

Alessandro Cipriani • Maurizio Giri

Música Electrónica y Diseño Sonoro

Teoría y Práctica con Max 8 • volumen 1

Temas tratados:

Síntesis y Procesamiento de sonido - Frecuencia, Amplitud y Forma de onda - Envolventes y glisandos - Síntesis Aditiva y síntesis Vectorial - Fuentes de ruido - Filtros - Síntesis Sustractiva - Realización de Sintetizadores Virtuales - Ecualizadores, Impulsos y cuerpos resonantes - Señales de control y LFO - Técnicas de programación con Max y MSP

"El libro de Alessandro Cipriani y Maurizio Giri es uno de los primeros cursos sobre música electrónica que integra explícitamente percepción, teoría y práctica, usando ejemplos de síntesis de sonido en tiempo real que pueden ser manipulados y personalizados. En mi opinión, Cipriani y Giri han realizado un trabajo maestro al permitir que los conocimientos experimental y teórico se refuercen entre sí. Este libro puede ser utilizado, ya sea como libro guía en un contexto académico, ya sea como un recurso para el autoaprendizaje. Adicionalmente, este libro incluye una introducción exhaustiva al procesamiento digital de señales con Max y es, además, una introducción maravillosa a los conceptos de programación en ese software. Espero que aproveches los excelentes ejemplos en Max que los autores han creado. Son esclarecedores y al mismo tiempo entretenidos, y suenan lo suficientemente bien como para ser usados en el escenario. También son dignos de examinar como modelos para tus propios patches en Max, o para extenderlos de nuevas maneras." (del prefacio por **David Zicarelli**)

Este es el primer libro de un método educativo integral compuesto por varios volúmenes. A cada capítulo de teoría corresponde un capítulo de práctica con el software Max (uno de los programas más potentes utilizados para procesamiento de sonido en tiempo real, tanto en Windows como en macOS) y una sección en línea: de esta manera, el estudiante adquiere habilidades y competencias teórico-prácticas de manera integral. El texto puede ser estudiado como autodidactas o bajo la guía de un docente. Por consiguiente, es ideal para quienes comienzan desde cero, aunque también es muy útil para quienes desean profundizar sus habilidades en el campo del diseño de sonido y de la música electrónica.

ALESSANDRO CIPRIANI es coautor del libro *Virtual Sound*, que trata sobre programación Csound. Sus composiciones han sido interpretadas en importantes festivales internacionales de música electrónica y publicadas por Computer Music Journal, International Computer Music Conference, CNI, Edi-Pan, Everglade, etc. Ha compuesto música para el Teatro de Ópera de Pekín, para películas, obras de teatro y documentales en los que el entorno sonoro, los diálogos y la música, procesados en la computadora, se fusionan, adquiriendo funciones mixtas. Ha realizado seminarios en numerosas universidades europeas y americanas (Universidad de California, Academia Sibelius Helsinki, Conservatorio Tchaikovsky de Moscú, DMU-Leicester, etc.). Es titular de la Cátedra de Composición Musical Electroacústica, del Conservatorio de Frosinone. Con su grupo *Edison Studio* ha compuesto 6 bandas sonoras en sorround 5.1 para películas de cine mudo, entre las cuales están: *La Corazzata Potemkin*, *Das Cabinet des Dr. Caligari* e *Inferno*, publicadas en DVD por la Cineteca di Bologna. Es miembro del Consejo Editorial de la revista Organized Sound (Cambridge University Press).

MAURIZIO GIRI es compositor, docente y desarrollador de software musical. Enseña Composición en el Conservatorio de Latina, y Técnicas de programación con Max en los conservatorios de Latina y Frosinone (Italia). Compone música instrumental y electroacústica. Ha desarrollado softwares para la composición algorítmica, la improvisación electroacústica y el live electronics. Es fundador de *Amazing Noises*, una empresa informática que implementa aplicaciones de música y plug-in para dispositivos móviles y computadoras tradicionales. Es partner de Ableton para el desarrollo de dispositivos Max for Live. Ha publicado tutorial sobre Max en revistas especializadas. Además, ha sido seleccionado como artista residente en París (Cité Internationale des Arts) y en Lyon (GRAME). Ha colaborado con el Institut Nicod alla École Normale Supérieure de París, y en un proyecto de filosofía del sonido.

ISBN 978-88-99212-17-9

www.contemponet.com