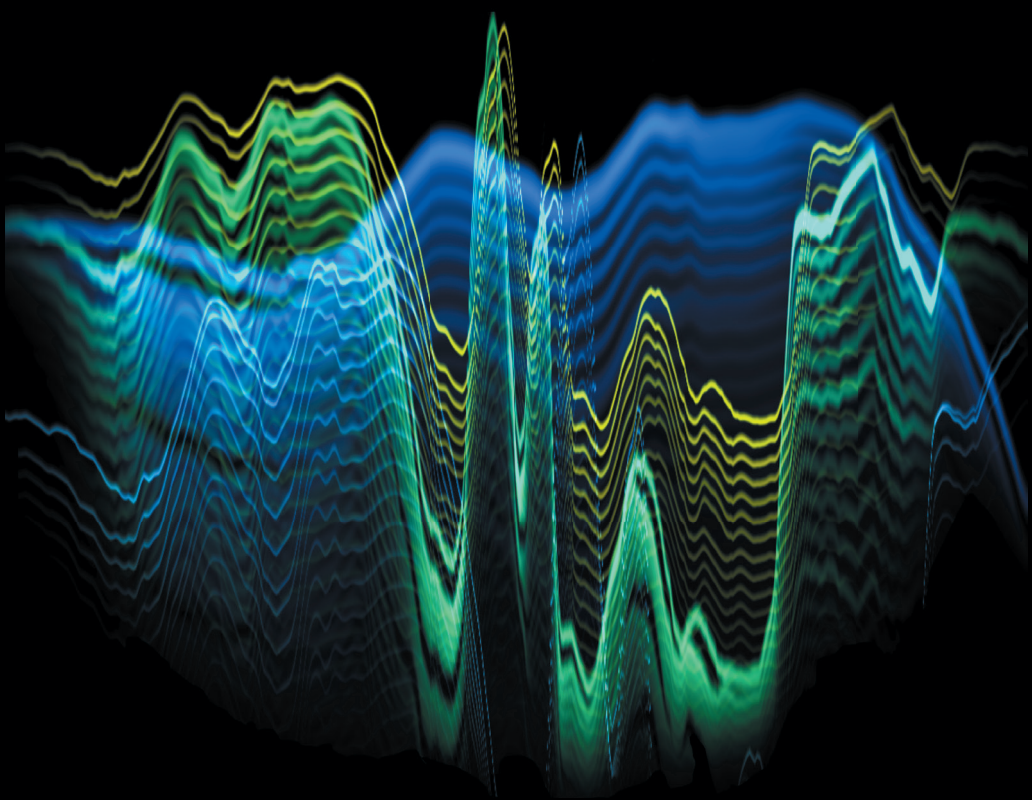
Alessandro Cipriani • Maurizio Giri

Musica Elettronica e Sound Design

Teoria e Pratica con **Max 7** • volume 1



ConTempoNet

Questo è un estratto del libro:

MUSICA ELETTRONICA E SOUND DESIGN

Teoria e Pratica con Max 7 - Volume 1

per maggiori informazioni:

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

CIPRIANI A. - GIRI M.
MUSICA ELETTRONICA e SOUND DESIGN
Teoria e Pratica con Max 7
Vol. 1
ISBN 978-88-99212-00-1

Prima edizione 2009
Seconda edizione 2013
Terza edizione 2016

Realizzazione figure: Gabriele Cappellani e Maurizio Refice
Realizzazione esempi interattivi: Francesco Rosati
Realizzazione indice analitico: Salvatore Mudanò
Consulenza glottodidattica: Damiano De Paola
Immagine di copertina: Federico Foderaro

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

INDICE

PREFAZIONE
di David Zicarelli

Potrà sembrarvi strano, ma molti anni fa, quando cercavo di imparare come si creano suoni al computer leggendo libri ed articoli, dovevo limitarmi ad immaginare quali suoni producessero le varie tecniche di sintesi. Anche se penso che la mia immaginazione ne sia stata stimolata, sono felice che nel frattempo la tecnologia si sia evoluta al punto che oggi quasi ogni tecnica di sintesi può essere realizzata in tempo reale con un normale computer. L'esperienza percettiva, infatti, è un elemento importantissimo nell'apprendimento delle tecniche di sintesi digitale.

Il libro di Alessandro Cipriani e Maurizio Giri costituisce uno dei primi corsi di musica elettronica che integra esplicitamente percezione, teoria e pratica, usando esempi di sintesi in tempo reale che si possono manipolare e personalizzare. Dal mio punto di vista, la manipolazione del suono costituisce un aspetto estremamente importante nell'apprendimento: consente di acquisire quella che Joel Chadabe chiama "conoscenza predittiva", cioè l'abilità di intuire ciò che succederà ad un suono prima che si compia un'azione per modificarlo. Tutti abbiamo una certa conoscenza predittiva: ad esempio quasi tutti sappiamo che, girando una manopola del volume in senso orario, il suono che viene dal nostro amplificatore aumenterà di intensità. Una volta che entriamo nel regno della sintesi digitale del suono, le cose si fanno molto più complicate di una manopola del volume, e abbiamo bisogno di fare esperienza diretta di manipolazione e di percezione per approfondire la nostra conoscenza predittiva.

Comunque, per istruirsi in modo completo sul suono prodotto digitalmente, abbiamo bisogno di molto di più che la semplice conoscenza predittiva. È necessario sapere perché le nostre manipolazioni generano i cambiamenti che percepiamo. Tale conoscenza teorica rinforza la nostra conoscenza esperienziale e intuitiva e, allo stesso tempo, la nostra esperienza fornisce significato percettivo alle spiegazioni teoriche.

Secondo il mio parere, Cipriani e Giri hanno compiuto un lavoro magistrale consentendo all'esperienza e alla conoscenza teorica di rinforzarsi l'una con l'altra. Questo libro funziona sia come testo all'interno di un corso, sia come mezzo per l'autoapprendimento. In aggiunta, il libro include un'introduzione completa all'elaborazione digitale dei segnali con Max e costituisce una splendida introduzione ai concetti di programmazione con questo software.

Come vedrete, i capitoli di teoria sono denominati "T", mentre la pratica e la conoscenza esperienziale sono incluse nei capitoli "P". Questi capitoli si alternano, come si muovono la gamba sinistra e quella destra quando si sale una scala, approfondendo e raffinando i concetti a livelli sempre più alti di sofisticazione.

Spero che trarrete vantaggio dagli eccellenti esempi di Max creati dagli autori: sono insieme divertenti e illuminanti, e suonano abbastanza bene da poter essere utilizzati anche sul palco. Vale la pena esaminarli come modelli per le

vostre patch Max, o per estenderli in modi sempre nuovi. Ma qualche minuto di "gioco" con gli esempi non è la stessa cosa che studiarli in rapporto ai concetti espressi nel libro. Il libro infatti fornisce il linguaggio per esprimere tali concetti in relazione ai fondamenti teorici. Conoscere la teoria è essenziale, perché presumibilmente state per leggere questo libro perché volete diventare persone che sanno fare molto più che girare una manopola.

Questo è sia l'augurio mio, sia quello degli autori. Voglio augurarvi buona fortuna in questa nuova avventura, e anche ringraziare i miei due amici italiani per aver creato una risorsa per l'apprendimento della musica digitale così completa, proprio quella che desideravo che esistesse quando ero uno studente!

David Zicarelli
Fondatore della Cycling'74, casa produttrice di Max

PREFAZIONE ALLA PRIMA EDIZIONE
di Alvise Vidolin

Il libro di Alessandro Cipriani e Maurizio Giri *Musica Elettronica e Sound Design* è un solido testo didattico che si rivolge alle persone desiderose di capire cosa sia la musica elettronica, partendo dall'esperienza diretta sul suono e sulle sue tecniche di sintesi, di elaborazione e di controllo che hanno segnato lo sviluppo di questa disciplina per farla diventare, come diceva Luciano Berio, «parte del pensare musicale di tutti i giorni.» È un libro in un certo senso controcorrente perché non rincorre gli "effetti speciali" preconfezionati che hanno decretato il successo commerciale di molti strumenti musicali elettronici: fornisce invece allo studente il metodo, gli strumenti teorici e la prassi operativa non solo per ottenere uno specifico "effetto", ma soprattutto per inventarne di originali e realizzati in funzione delle proprie esigenze musicali.
Gli autori trasformano in metodo didattico la filosofia dei così detti *strumenti musicali aperti* che ha caratterizzato la nascita della musica elettronica. Questa tipologia di strumenti consentiva un nuovo sistema di produzione musicale, basato su elementi "componibili", mediante il quale il compositore poteva creare in maniera completa la sua musica, fino alla realizzazione sonora definitiva, che veniva memorizzata su nastro magnetico per le audizioni acusmatiche in concerto. Purtroppo le tecnologie analogiche dei pionieri della musica elettronica non consentivano di realizzare dal vivo la complessità musicale richiesta dai compositori e si dovette attendere lo sviluppo dell'informatica in tempo reale per vedere realizzato questo desiderio. Uno dei programmi che ha efficacemente contribuito a questo sviluppo è stato Max, sviluppato all'IRCAM da Miller Puckette verso la metà degli anni '80. Fin dalle prime versioni Max ha fornito i mezzi per un libero controllo dal vivo di qualsiasi dato MIDI e in quelle successive, con l'aggiunta di MSP, il controllo si è esteso ai segnali audio e in tempi più recenti, con l'avvento di Jitter, anche ai segnali video. La caratteristica che rende Max particolarmente efficace per la realizzazione di qualsiasi produzione musicale, sia in laboratorio, sia *live*, è la sua concezione aperta. L'apertura sta nel fatto che non fornisce ambienti musicali predefiniti, che inevitabilmente invecchiano con l'avvento delle nuove mode, bensì fornisce gli oggetti per costruire l'ambiente musicale richiesto dalla composizione che si vuole creare e/o eseguire. E non è un caso che Cipriani e Giri abbiano scelto proprio Max come strumento operativo per le esercitazioni pratiche che affiancano in maniera costante ed approfondita le parti teoriche del libro, ampliandone le potenzialità con librerie di abstraction scaricabili dal sito del libro assieme ad altri utili materiali.
Ovviamente i musicisti che sono abituati a scegliere un nuovo strumento elettronico in base ai risultati musicali immediati che questo offre, resteranno delusi da tale approccio, ma se avranno la pazienza di ampliare le proprie conoscenze sul suono, di entrare nella logica della programmazione ad oggetti e di lavorare cercando di raggiungere un obiettivo piuttosto che affidarsi alla scelta di effetti sonori preconfezionati, scopriranno le infinite potenzialità offerte dai programmi informatici aperti di cui Max è un valido esempio.
Questa concezione aperta è un punto cruciale per l'apprendimento della musica elettronica: è il metodo che consente a tale disciplina di inserirsi a pieno titolo

all'interno della tradizione didattica della musica, affiancandosi ai metodi analitici e alle prassi esecutive dei corsi tradizionali di composizione e di strumento. D'altra parte lo studio della musica non può prescindere dallo studio del suono nelle sue varie dimensioni, integrando la ricca tradizione acustica con i nuovi saperi e i diversi metodi della musica elettronica. In altri termini il musicista deve studiare il suono non solo come dato teorico astratto, ma deve soprattutto assimilarlo attraverso l'esperienza sensibile, mettendo continuamente a confronto l'elemento teorico e l'esperienza percettivo-musicale.
Questo libro di *Musica Elettronica e Sound Design* è proprio lo strumento didattico ideale per le nuove generazioni di musicisti, in quanto riesce sempre a creare un perfetto equilibrio fra saperi teorici e realizzazioni pratiche. L'opera, articolata in tre volumi, adotta un metodo didattico organico con una concezione aperta e interattiva dell'insegnamento che si rivela efficace sia per una didattica gestita dal docente sia per l'autoapprendimento. Per molte esercitazioni pratiche gli autori trasformano Max in un completo "laboratorio di liuteria elettronica", partendo dai primi suoni dell'elettronica analogica, per approfondire le principali tecniche di sintesi e di elaborazione dei suoni, realizzando strumenti virtuali e di interazione, programmando controlli gestuali per l'esecuzione dal vivo, creando sistemi di diffusione e di spazializzazione per l'ascolto. La didattica diventa in questo modo interattiva in quanto il laboratorio virtuale funziona in tempo reale e consente di ascoltare passo dopo passo il processo realizzativo, verificando puntualmente il proprio operato.
In conclusione questo libro presenta tutte le caratteristiche per diventare il testo di riferimento dei corsi di musica elettronica dei Conservatori italiani e non solo, proseguendo il cammino iniziato con successo da *Il suono virtuale*, scritto sempre da Alessandro Cipriani a quattro mani con il compianto Riccardo Bianchini.

Alvise Vidolin
Venezia 08/09/2009

INTRODUZIONE

Questo è il primo volume (ora aggiornato a Max 7) di una serie di 3 volumi sulla sintesi e l'elaborazione digitale del suono. Il piano dell'opera prevede anche:

- un secondo volume che tratta diversi temi fra cui l'audio digitale, i processori di dinamica, le linee di ritardo, il protocollo MIDI e il tempo reale e un primo capitolo dedicato all'arte dell'organizzazione del suono;
- un terzo volume che concerne riverbero e spazializzazione, le diverse tecniche di sintesi non lineare (come AM, FM, waveshaping e tecniche di distorsione del suono), la sintesi granulare, l'analisi e risintesi, i modelli fisici, il sound design procedurale e un secondo capitolo dedicato all'organizzazione del suono.

LIVELLO RICHIESTO

Tutti i volumi alternano parti teoriche a sezioni di pratica al computer, che vanno studiate in stretta connessione. Questo primo volume può essere utilizzato da utenti di diverso livello di preparazione.

Il livello minimo richiesto per chi inizia a studiare il Vol.1 comprende:

- i primi rudimenti di teoria musicale (note, scale, accordi etc.)
- una competenza di base nell'utilizzo di un computer (saper salvare un file, copiare, cancellare etc.).

Il testo va studiato alternando ogni capitolo di teoria a quello corrispettivo di pratica incluse le attività al computer. La parte teorica non è sostitutiva di testi teorici sulla sintesi. Si tratta, invece, di un indispensabile compendio teorico al lavoro pratico di programmazione e di invenzione di suoni al computer, ed è parte quindi di un sistema didattico organico. Il percorso di questo volume può essere svolto in auto-apprendimento oppure sotto la guida di un insegnante.

I TEMPI DI APPRENDIMENTO

I tempi di apprendimento, come è ovvio, sono diversi da persona a persona. In particolare daremo conto di tempi di mero riferimento nelle due modalità: auto-apprendimento e apprendimento sotto la guida di un docente esperto.

Auto-apprendimento (300 ore globali di studio individuale)

Capitoli	Argomento	Totale ore
1T+1P+IA	Sintesi del suono	100
2T+2A	Sintesi Additiva	60
3T+3P+IB	Sottrattiva e filtri	110
4T+4P	Segnali di Controllo	30

Apprendimento con docente
(corso di 60 ore in classe + 120 di studio individuale)

Capitoli	Argomento	Lezioni	Feedback	Studio	Totale ore
1T+1P+IA	Sintesi del suono	16	4	40	60
2T+2P	Sintesi Additiva	10	2	24	36
3T+3P+IB	Sottrattiva e filtri	18	4	44	66
4T+4P	Segnali di controllo	5	1	12	18

GLI ESEMPI INTERATTIVI

Il percorso della parte teorica è accompagnato da molti esempi interattivi reperibili sul sito www.**********. Utilizzando questi esempi, si può fare esperienza immediata del suono e della sua creazione ed elaborazione senza aver ancora affrontato alcun lavoro pratico di programmazione. In questo modo lo studio della teoria è sempre in connessione con la percezione del suono e delle sue possibili modificazioni. Far interagire percezione e conoscenza nello studio del sound design e della musica elettronica è stato da sempre un nostro obiettivo, e questo criterio guida l'intera opera didattica, comprensiva anche di ulteriori materiali online che verranno man mano aggiornati ed ampliati.

TEORIA E PRATICA

L'impostazione didattica è basata proprio sull'interazione (per noi imprescindibile) fra teoria e pratica. Uno dei problemi nel campo dell'elaborazione del suono, infatti, è quello di avere esperti di teoria che normalmente non si trovano ad affrontare problemi concreti riguardanti la pratica dell'invenzione del suono, e persone (molto più numerose) che amano lavorare al computer con i suoni, ma che spesso hanno una scarsa coscienza tecnico-teorica di cosa stiano facendo, e una scarsa capacità di modificare ciò che i software che utilizzano li "costringono" a fare. Il mercato propone sempre più oggetti tecnologici meravigliosi, ma difficili da personalizzare. Un'informazione spesso approssimativa e poco sistematica, unita alla rapida obsolescenza dei sistemi, contribuisce a mantenere gli utenti in una (apparentemente piacevole) ignoranza, e quindi in una condizione di scarsa libertà, costringendoli, in un certo senso, ad usare le macchine e il software che acquistano in modo superficiale e ad aggiornarle continuamente spesso senza averne compreso la natura profonda. In questo senso intraprendere lo studio di questo libro significa anche iniziare ad acquisire una maggiore consapevolezza dell'uso dei software commerciali di sintesi ed elaborazione del suono.

L'IMPOSTAZIONE DIDATTICA

Sulla base dei concetti appena esposti, abbiamo pensato di colmare il vuoto di informazione riguardante questa materia, avanzando nella direzione intrapresa da Cipriani e Bianchini con il testo "Il Suono Virtuale", dedicato alla sintesi ed elaborazione del suono. La differenza con quel testo è grande, sia per la qualità degli esempi proposti, sia perché l'impostazione didattica è completamente diversa. Esiste pochissima bibliografia sulla metodologia didattica della musica elettronica. A questo scopo abbiamo riflettuto sulla possibilità di approfondire questa tematica e progettare finalmente un sistema didattico organico, mutuando alcune idee e tecniche dalla didattica delle lingue straniere, in modo da sviluppare una concezione più aperta e interattiva dell'insegnamento e dell'apprendimento.

Per questo abbiamo inserito, oltre agli esempi interattivi, anche contratti formativi per ogni capitolo, attività di ascolto e analisi, test, glossari, indicazioni discografiche e introduzioni storiche (online) oltre a tante altre novità contenute nei capitoli di pratica come le attività di sostituzione di parti di

algoritmi, correzione, completamento e analisi di algoritmi, costruzione di nuovi algoritmi, compiti di reverse engineering (cioè, a partire dall'ascolto di un suono, cercare di inventare un algoritmo che possa creare un suono simile a quello ascoltato). Inoltre, all'indirizzo **www.********** è possibile scaricare un documento PDF contenente attività pratiche aggiuntive. Nel corso dei capitoli la presenza di tali attività viene segnalata con l'icona visibile qui a lato.

Il sistema, composto da 3 volumi e una sezione online è multi-piattaforma, e la teoria è costruita in modo tale da poter fare da base a possibili altri testi di pratica basati su software diversi, utilizzando lo stesso percorso didattico.

Max

La parte pratica del libro è basata sul software Max. Questo programma, scritto originariamente da Miller Puckette, è stato sviluppato ed esteso da David Zicarelli, ed è prodotto dalla sua società Cycling '74 (www.cycling74.com).
Max è un ambiente grafico interattivo per la musica, l'audio e il multimedia. È usato in tutto il mondo da musicisti, compositori, sound designer, artisti multimediali etc. ed è diventato, di fatto, uno standard per la creatività tecnologicamente evoluta in ambito musicale e visivo.
È un linguaggio di programmazione interamente grafico, ed è quindi relativamente facile da apprendere pur essendo molto potente.
In Max si creano programmi connettendo tra loro degli oggetti grafici. Questi oggetti possono eseguire dei calcoli, produrre o elaborare suoni, creare immagini, o essere usati come interfaccia grafica. Si possono così realizzare sintetizzatori, campionatori, riverberi, effetti, e molto altro.
In pratica viene adottata la metafora del synth modulare: ciascun modulo svolge una particolare funzione e passa le informazioni ai moduli a cui è connesso. La differenza è che con Max si può lavorare ad un livello di dettaglio impensabile per un sintetizzatore già pronto per l'uso (hardware o software che sia).

INDICAZIONI PRATICHE

A corredo di questo libro sono stati realizzati molti materiali assolutamente indispensabili per procedere nell'apprendimento: esempi interattivi, *patch* (ovvero programmi scritti in Max), *sound file*, estensioni di libreria e altri materiali di supporto si trovano tutti all'indirizzo **www.**********.

Esempi Interattivi

Durante lo studio della teoria, prima di affrontare la parte pratica, è importante utilizzare gli esempi interattivi. Lavorare con questi esempi sarà di notevole aiuto per affrontare poi la parte pratica relativa all'argomento trattato.

File di esempio

I file di esempio (*patch*) sono utilizzabili con il software Max, scaricabile dal sito ufficiale www.cycling74.com.

Alternanza di Teoria e Pratica

Nel libro i capitoli di teoria si alternano ai capitoli di pratica. Il lettore si troverà quindi ad affrontare tutto un capitolo di teoria per poi passare al corrispettivo

capitolo di pratica (ad esempio tutto il capitolo 1T e poi tutto il capitolo 1P). In alternativa può scegliere di leggere un paragrafo di teoria e subito dopo il paragrafo corrispondente di pratica per poi passare al paragrafo successivo (ad esempio 1.1T e 1.1P, poi 1.2T e 1.2P etc.).

Gli Interludi
Da notare che fra il primo e il secondo capitolo, e fra il terzo e il quarto capitolo ci sono 2 "interludi" tecnici, rispettivamente l'Interludio A e l'Interludio B, dedicati specificamente al linguaggio Max e non legati ai temi trattati nella teoria, ma ugualmente necessari per procedere nel percorso tracciato nel libro. Dopo aver affrontato la teoria e la pratica del primo capitolo, prima di passare al secondo capitolo è fondamentale studiare l'interludio A. Ciò vale, ovviamente anche per l'interludio B, da studiare subito dopo aver completato i capitoli 3T e 3P.

L'apprendimento di Max
L'apprendimento di Max (e in generale della sintesi ed elaborazione del suono) richiede applicazione e concentrazione. Al contrario di molti software commerciali, infatti, Max consente una flessibilità massima nella programmazione, e quindi consente una grande libertà a chi programma gli algoritmi; ma per poter usufruire di questa libertà è fondamentale evitare di saltare i passaggi consigliati nel libro e procedere in modo sistematico. Un apprendimento "intuitivo" o a salti dà scarsi risultati in Max, specialmente all'inizio del percorso di apprendimento. Questo software è un vero e proprio "strumento musicale" e va studiato come si studierebbe uno strumento tradizionale (ad esempio un violino); è necessario cioè utilizzarlo con continuità, partendo dagli esercizi di base e affrontando via via le tecniche più complesse per evitare di dimenticare le conoscenze e di perdere le abilità acquisite. Solo così sarà possibile arrivare a una vera padronanza del programma.

Bibliografia e sitografia
Si è scelto di inserire nel testo soltanto una bibliografia assolutamente essenziale, e i riferimenti bibliografici relativi ai testi citati nel libro. Una bibliografia più completa e una sitografia è disponibile online.

Prima di cominciare
Per iniziare a lavorare con questo testo è necessario scaricare il programma **Esempi Interattivi** che si trova alla pagina di supporto www.**********.
Durante la lettura dei capitoli di teoria si farà costante riferimento agli esempi contenuti in questa applicazione.
Per affrontare la parte pratica è invece necessario aver installato il programma **Max**, reperibile al sito www.cycling74.com. Bisogna inoltre scaricare la libreria **Virtual Sound Macros** dalla pagina di supporto di questo testo (www.**********.); nella stessa pagina troverete istruzioni dettagliate sulla procedura da seguire per la corretta installazione della libreria.
Sempre a partire dalla pagina di supporto troverete le *patch* (programmi Max) relative a tutti i capitoli di pratica e i file audio per gli esercizi di *reverse engineering*.

Commenti e segnalazioni
Correzioni e commenti sono benvenuti. Vi preghiamo di inviarli per e-mail a:
a.cipriani@edisonstudio.it oppure maurizio@giri.it
http://www.edisonstudio.it/alessandro-cipriani/
http://www.giri.it

RINGRAZIAMENTI

Si ringraziano:
Gabriele Cappellani, Salvatore Mudanò e Francesco "Franz" Rosati per il loro paziente e lungo lavoro;
Eugenio Giordani, Giuseppe Emanuele Rapisarda, Fausto Sebastiani e Alvise Vidolin per la loro disponibilità.

DEDICA

Questo testo è dedicato a Riccardo Bianchini, che avrebbe voluto realizzare anche quest'opera didattica, ma che purtroppo è prematuramente scomparso prima che il lavoro iniziasse. Abbiamo raccolto alcuni suoi materiali (anche inediti), li abbiamo editati e citati in alcuni paragrafi di teoria. Questo è stato un modo, idealmente, per avere Riccardo ancora con noi. Un ringraziamento particolare va ad Ambretta Bianchini, per la grande disponibilità e sensibilità dimostrataci in questi anni di lavoro.

Buona lettura,
Alessandro Cipriani e Maurizio Giri

LEGENDA DEI SIMBOLI UTILIZZATI

• ATTIVITÀ ED ESEMPI INTERATTIVI

• ATTIVITÀ PRATICHE AGGIUNTIVE DISPONIBILI ONLINE

• COMPITI UNITARI

• CONCETTI DI BASE

• DETTAGLI TECNICI

• VERIFICA

1T
INTRODUZIONE ALLA SINTESI DEL SUONO

CONTRATTO FORMATIVO

PREREQUISITI PER IL CAPITOLO

- Conoscenze di base degli strumenti informatici (operazioni base, gestione delle cartelle, scheda audio etc.)
- Conoscenza minima della teoria musicale (semitoni, ottave, tempi etc.)

OBIETTIVI

Conoscenze

- Conoscere i percorsi mediante i quali si realizza la sintesi e l'elaborazione del suono
- Conoscere i parametri principali del suono e le loro caratteristiche
- Conoscere le codifiche dell'altezza e dell'intensità
- Conoscere i rapporti fra gli intervalli musicali nei diversi sistemi di accordatura
- Conoscere i diversi formati dei file audio

Abilità

- Saper individuare all'ascolto mutamenti di frequenza e d'ampiezza e saperne descrivere le caratteristiche
- Saper individuare le varie fasi dell'inviluppo di un suono o di un suono glissato

CONTENUTI

- Sintesi ed elaborazione del suono al computer
- Timbro, altezza e intensità di un suono: (teoria)
- Glissando e inviluppo d'ampiezza: (teoria)
- Rapporti tra frequenze, altezze e codifiche MIDI
- Uso di suoni campionati (cenni)

TEMPI - Cap.1 (teoria e pratica) + Interludio A

Autodidatti

Per 300 ore globali di studio individuale (vol. I, teoria e pratica):

- ca. 100 ore

Corsi

Per un corso globale di 60 ore in classe + 120 di studio individuale (vol. I, teoria e pratica):

- ca. 16 ore frontali + 4 di feedback
- ca. 40 di studio individuale

ATTIVITÀ

- Esempi interattivi

VERIFICHE

- Test a risposte brevi
- Test con ascolto e analisi

SUSSIDI DIDATTICI

- Concetti di base - Glossario - Un po' di storia

1.1 SINTESI ED ELABORAZIONE DEL SUONO

L'introduzione dell'elettronica e, soprattutto, del computer nella musica ha consentito a compositori e musicisti di gestire e manipolare i suoni con una precisione e una libertà impensabili con i soli mezzi acustici.
Grazie all'uso del computer è infatti possibile modellare il suono in ogni modo immaginabile; si dice spesso che mentre il compositore "tradizionale" compone con i suoni, il compositore elettronico compone i suoni, ovvero entra nel suono, nelle sue componenti elementari, creandole e trasformandole a suo piacimento.
La stessa cosa avviene, per fare un parallelo, nella grafica e nell'animazione: grazie al computer è possibile creare immagini e sequenze filmate estremamente realistiche, che sarebbe difficile produrre con altri mezzi. Attualmente quasi tutti gli effetti speciali al cinema sono realizzati con il computer, e sempre più spesso i personaggi virtuali "recitano" al fianco di attori in carne ed ossa.
Il "segreto" di questa flessibilità sta nel passaggio dal mondo analogico (quello degli oggetti concreti) a quello digitale (ovvero dei numeri): il processo di digitalizzazione consiste appunto nel trasformare un'informazione (un testo, un suono, un'immagine) in numeri.[1] Una volta che un'immagine o un suono sono stati convertiti in una sequenza numerica, possono subire qualunque tipo di trasformazione, perché i numeri, grazie a secoli di sviluppo delle tecniche matematiche, possono essere trasformati e manipolati in qualsiasi modo.

Questo testo si concentrerà essenzialmente su due aspetti: la sintesi e l'elaborazione del suono.

- La **sintesi del suono** (sound synthesis) si riferisce alla generazione elettronica di un suono. In pratica si tratta della possibilità di creare un suono sulla base di alcuni parametri scelti in funzione del risultato sonoro che si vuole ottenere.

- L'**elaborazione del suono**, o del segnale, (signal processing) si riferisce ai processi utilizzati per modificare un suono già esistente, ad esempio un suono di una chitarra che abbiamo precedentemente registrato, un suono generato con una particolare tecnica di sintesi, etc.

SINTESI DIGITALE DEL SUONO

Per ottenere qualsiasi tipo di suono utilizzando un linguaggio di programmazione per la sintesi e l'elaborazione del suono, scriveremo nel computer le informazioni sul tipo di "macchina virtuale" che vogliamo costruire (realizzeremo cioè un **algoritmo**[2]) e le operazioni che questa macchina deve compiere.

---

[1] Approfondiremo questo concetto nel corso del capitolo.

[2] Un algoritmo è un procedimento che comporta una serie ordinata di istruzioni elementari: queste istruzioni, eseguite in successione, permettono di risolvere un problema, di ottenere un risultato. Informalmente possiamo dire che anche una ricetta di cucina è un algoritmo; si tratta infatti di una serie di istruzioni che danno come risultato una pietanza. In informatica un algoritmo è una sequenza di istruzioni scritta in un particolare linguaggio di programmazione che permette al computer di svolgere un compito definito.

Una volta scritte queste istruzioni, chiederemo al programma (Max o altri) di eseguirle e di creare un flusso di dati numerici in cui sono rappresentate digitalmente[3] tutte le caratteristiche del suono o dei suoni che abbiamo richiesto. Tra la generazione di questo flusso di dati digitali e l'ascolto del suono avviene un'altra operazione fondamentale che richiede una **scheda audio**. La scheda legge i dati digitali e li trasforma in segnale elettrico che viene inviato all'amplificatore e poi agli altoparlanti. In questo caso la scheda opera una conversione da digitale ad analogico (D/A), cioè ci consente di ascoltare dei suoni le cui caratteristiche sono scritte in un flusso di dati digitali (fig. 1.1).

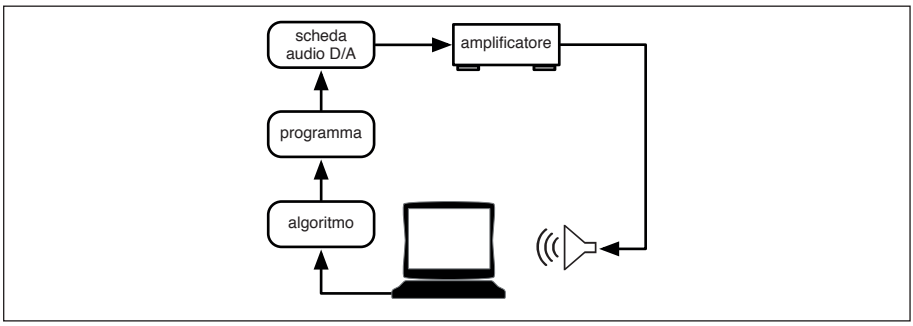


fig. 1.1: sintesi in tempo reale

Questi dati possono anche essere memorizzati in un file audio che verrà salvato nel nostro hard disk, per permettere una riesecuzione dei dati stessi o una loro elaborazione. Quando il flusso dei dati è direttamente inviato alla scheda audio man mano che viene calcolato si ha una **sintesi in tempo reale** (*real time*). Quando invece il processo di calcolo viene prima svolto per intero (e memorizzato in un file audio) e solo successivamente inviato alla scheda audio per l'ascolto si ha una **sintesi in tempo differito** (*non-real time* o *offline*), vedi fig. 1.2.

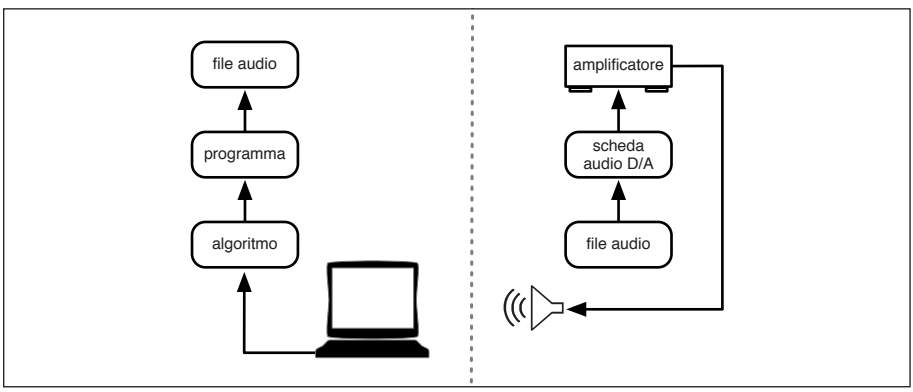


fig. 1.2: sintesi in tempo differito e successivo ascolto

[3] Ovvero sotto forma di numeri.

ELABORAZIONE DEL SUONO

L'elaborazione del suono consiste nella modifica di un suono preesistente, che può provenire sia da una fonte live, sia da un file audio. È possibile operare sia in tempo reale sia in tempo differito, in diversi modi. Vediamo tre possibilità:

1) SUONO PREESISTENTE IN TEMPO DIFFERITO,
ELABORAZIONE IN TEMPO DIFFERITO

Un suono di flauto, ad esempio, può essere registrato (con un microfono collegato alla scheda audio che opererà una conversione analogico-digitale[4]) su un file audio. Possiamo creare un algoritmo in cui specificheremo come quel file audio deve essere modificato, poi il programma eseguirà quei comandi e creerà un nuovo file audio che conterrà un suono di flauto, elaborato dal computer secondo le nostre indicazioni. Infine potremo ascoltare questo nuovo sound file operando una conversione digitale-analogica (fig. 1.3).

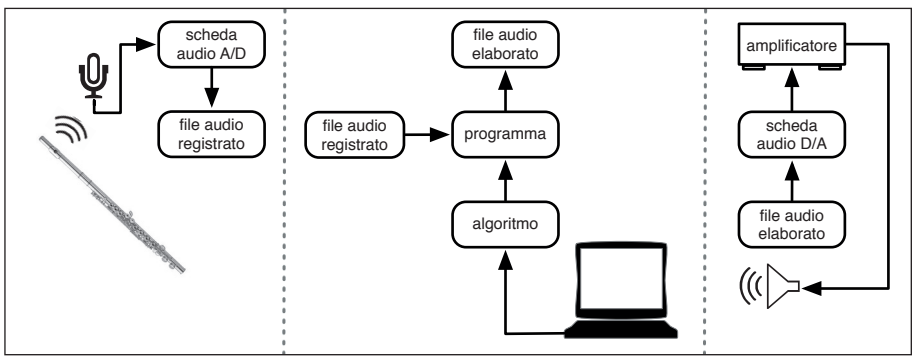


fig. 1.3: esempio di elaborazione in tempo differito

2) SUONO PREESISTENTE IN TEMPO DIFFERITO,
ELABORAZIONE IN TEMPO REALE

Il suono, come nell'esempio 1, proviene da un file audio. Il programma di elaborazione, eseguiti i comandi, invia il flusso di dati contenenti il suono elaborato direttamente alla scheda audio per l'ascolto in tempo reale. Oltre a ciò il programma può registrare, sempre in tempo reale, il risultato dell'elaborazione su un file audio (fig. 1.4).

3) SUONO IN TEMPO REALE,
ELABORAZIONE IN TEMPO REALE

Il suono proviene da una fonte live. Come nell'esempio precedente, il programma di elaborazione, eseguiti i comandi, invia il flusso di dati contenenti il suono elaborato direttamente alla scheda audio.

4 Ovvero trasformerà un suono reale in una sequenza di numeri.

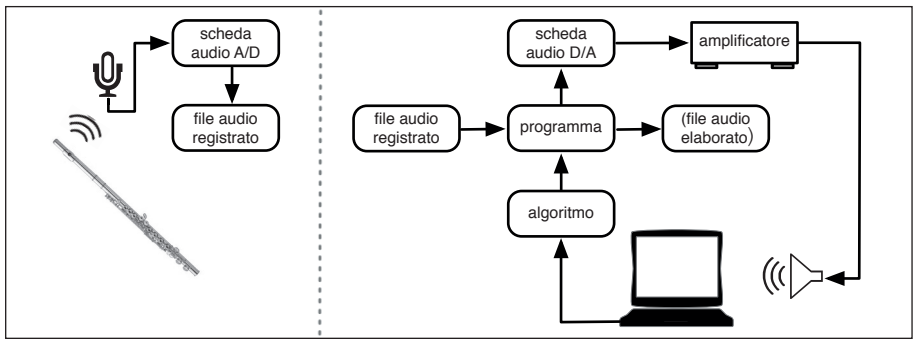


fig. 1.4: esempio di elaborazione in tempo reale da suono preesistente

Naturalmente, anche in questo caso il programma può registrare, sempre in tempo reale, il risultato dell'elaborazione su un file audio (vedi fig. 1.5).

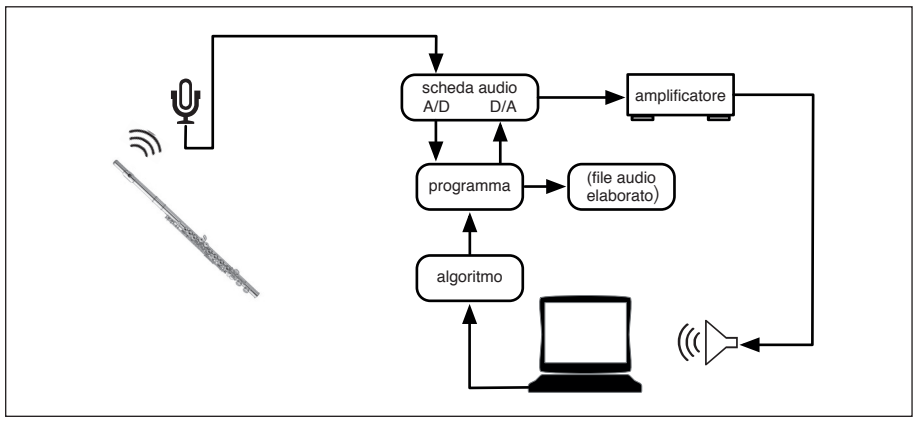


fig. 1.5: esempio di elaborazione in real-time

Definiamo **sistema DSP** l'insieme delle risorse hardware e software (scheda audio, linguaggio di programmazione etc.) che ci permette di elaborare e/o sintetizzare digitalmente un suono (o segnale). Il termine **DSP** è un acronimo che sta per Digital Signal Processing (Elaborazione Digitale del Segnale).

TEMPO REALE - TEMPO DIFFERITO

Abbiamo visto che sia la sintesi sia l'elaborazione del suono possono avvenire in tempo reale o in tempo differito. A prima vista il metodo più vantaggioso appare senz'altro il tempo reale, che ci fornisce un feedback istantaneo e ci consente di valutare immediatamente l'efficacia dell'algoritmo che si sta mettendo a punto e a cui si possono fare le opportune modifiche e migliorie.
A cosa serve quindi il tempo differito?
- Innanzitutto a realizzare degli algoritmi che il computer non è in grado di eseguire in tempo reale: se ad esempio per sintetizzare o elaborare un suono che dura 1 minuto il computer impiega 2 minuti, si dovrà per forza registrare il risultato su disco per poterlo ascoltare una volta che il processo di sintesi o elaborazione sia finito.

Agli albori della computer music tutti i processi di sintesi ed elaborazione del suono erano realizzati in tempo differito, perché un calcolatore non aveva abbastanza potenza per il tempo reale. Con l'aumentare della potenza dei calcolatori, è diventato possibile realizzare alcuni processi direttamente in tempo reale, e nel corso degli anni le capacità di un personal computer di realizzare algoritmi di sintesi ed elaborazione in tempo reale sono aumentate enormemente. Ma naturalmente, per quanto potenti possano diventare i calcolatori, sarà sempre possibile immaginare un processo talmente complesso da richiedere il tempo differito.
- Esiste poi una seconda categoria di processi che sono concettualmente in tempo differito, indipendentemente dalla potenza di calcolo dell'elaboratore: poniamo ad esempio di voler realizzare un algoritmo che, data una sequenza musicale suonata da uno strumento, scomponga tale sequenza in singole note e poi le riordini dalla più grave alla più acuta.
Per realizzare questo algoritmo abbiamo bisogno della sequenza completa, molto probabilmente registrata in un file audio, in modo che il computer la possa analizzare nel suo complesso e individuare la nota più grave e via via le note successive.
Questa analisi può ovviamente avvenire solo in tempo differito, dopo l'esecuzione: l'unico computer che potrebbe realizzare questo algoritmo in tempo reale (cioè mentre lo strumento sta suonando) è un computer in grado di prevedere il futuro!
- Un altro motivo per cui si ricorre al tempo differito è per risparmiare tempo. Contrariamente a quello che si può pensare, il tempo reale non corrisponde alla massima velocità di elaborazione possibile. Immaginiamo ad esempio di dover modificare, con una particolare tecnica di elaborazione, un file di suono della durata di 10 minuti: se la modifica avviene in tempo reale impiegherà ovviamente 10 minuti. Immaginiamo però che il nostro computer sia abbastanza potente da poter realizzare questa elaborazione, in tempo differito, in un minuto. Questo significa che il computer può eseguire i calcoli, per quella particolare tecnica di elaborazione, ad una velocità 10 volte superiore al tempo reale, ed è quindi conveniente ricorrere al tempo differito.

1.2 FREQUENZA, AMPIEZZA E FORMA D'ONDA

Frequenza, ampiezza e forma d'onda sono tre parametri fondamentali del suono. Ognuno di questi parametri influenza nell'ascoltatore la percezione sonora, in particolare:

a) la possibilità di distinguere un suono grave da uno acuto (frequenza)
b) la possibilità di distinguere un suono di forte intensità da uno di intensità minore (ampiezza)
c) la possibilità di distinguere diversi timbri (forma d'onda)[5]

[5] Vedremo più avanti come il parametro del timbro dipenda in realtà da diversi fattori concomitanti.

Vediamo una tabella (tratta da Bianchini, R., 2003) delle corrispondenze fra caratteristiche fisiche del suono, parametri musicali e sensazione sonora.

CARATTERISTICA	PARAMETRO MUSICALE	SENSAZIONE
Frequenza	Altezza	Acuto ↔ Grave
Ampiezza	Intensità	Forte ↔ Piano
Forma d'onda	Timbro	(Chiaro ↔ Scuro Armonico ↔ Inarmonico etc.)

TABELLA A : corrispondenza fra caratteristiche del suono, parametri musicali e sensazione sonora

FREQUENZA

La **frequenza** è il parametro fisico che determina l'altezza di un suono, cioè la caratteristica che consente di distinguere un suono acuto da un suono grave. La gamma delle frequenze udibili dall'uomo si estende da circa 20 a circa 20000 Hertz, cioè da 20 a 20000 cicli al secondo (spiegheremo tra un momento di cosa si tratta): al di sotto della minima frequenza percepibile, sotto i 20 cicli al secondo, si hanno gli infrasuoni, al di sopra di quella massima, sopra i 20000 cicli al secondo, si hanno gli ultrasuoni.[6] Se ci concentriamo sul campo delle frequenze udibili, quindi dei suoni, potremo affermare che maggiore è la frequenza, tanto più acuto sarà il suono.
Ma cosa intendiamo per Hertz o "cicli al secondo"? Per saperlo facciamo riferimento alla definizione di suono data da Riccardo Bianchini:

"Per suono si intende quel fenomeno meccanico dato da una perturbazione di un mezzo di trasmissione (in genere l'aria) che abbia caratteristiche tali da essere percepito dall'orecchio umano.[7] La vibrazione viene trasmessa all'aria, per esempio da una corda vibrante (vedi fig. 1.6). La corda si sposta avanti e indietro, e durante questo spostamento comprime le particelle d'aria (molecole) da un lato e le espande dall'altro. Successivamente il moto si inverte, e le molecole che prima erano state compresse si espandono e viceversa.
Le compressioni e le espansioni (cioè le perturbazioni dell'aria che inizialmente era in stato di quiete) si propagano poi con una certa velocità attraverso l'aria

6 In realtà la massima frequenza udibile diminuisce con l'età.

7 Ci sono molte teorie sulla natura del suono: Roberto Casati e Jérôme Dokic sostengono che l'aria è un mezzo attraverso cui il suono si trasmette, ma che il suono in sé è un evento localizzato nel corpo risonante, ovvero nel sistema meccanico che produce la vibrazione. (Casati, R., Dokic, J. 1994). Un altro punto di vista è quello espresso da Frova: "con il termine «suono» si dovrebbe intendere la sensazione, com'essa si manifesta a livello cerebrale, di una perturbazione di natura meccanica, a carattere oscillatorio, che interessa il mezzo interposto tra sorgente e ascoltatore" (Frova, A., 1999, pag.4).

circostante in tutte le direzioni, dando luogo a onde sferiche. Inizialmente la densità delle molecole d'aria è costante, cioè in ogni unità di volume (per esempio in un cm^3) vi è lo stesso numero di molecole.

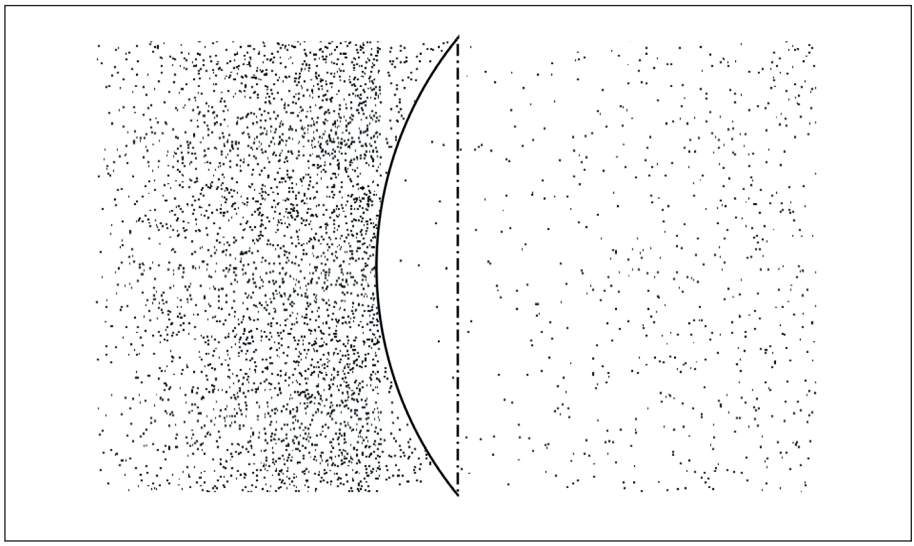


fig. 1.6: vibrazione di una corda

Questa densità può essere espressa da un valore di pressione. Quando l'aria viene perturbata, il valore di pressione non è più costante, ma varia da punto a punto: aumenta dove le molecole sono compresse, diminuisce dove le molecole sono espanse (vedi fig. 1.7).

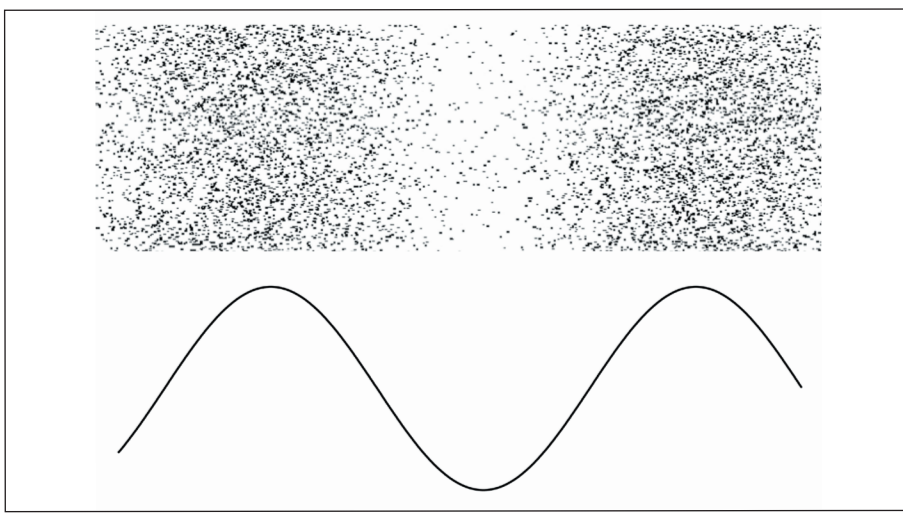


fig.1.7: compressione e rarefazione delle molecole dell'aria

Il fenomeno può essere studiato sia dal punto di vista dello spazio (osservando il valore della pressione nei vari punti in un determinato istante) sia dal punto

di vista del tempo (misurando il valore della pressione in uno stesso punto in funzione del tempo). Ad esempio, se immaginiamo di trovarci in un determinato punto, assisteremo a una successione di compressioni ed espansioni dell'aria (fig. 1.8).

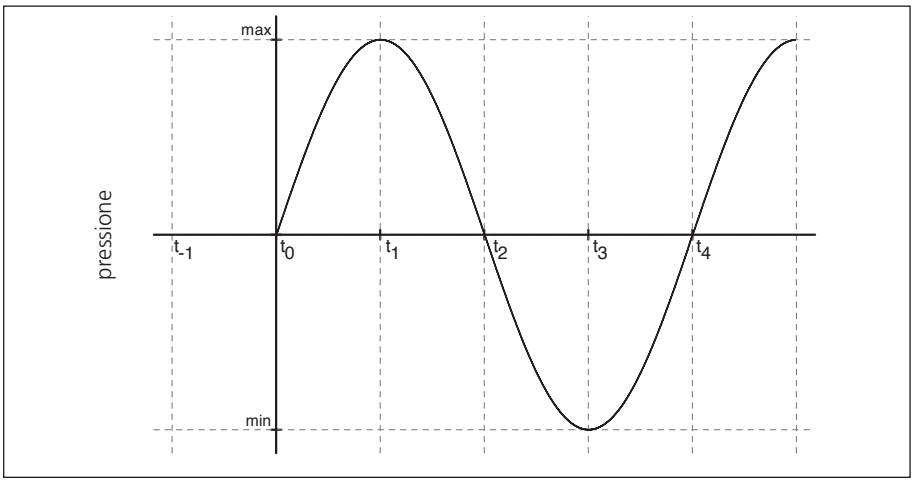


fig.1.8: rappresentazione grafica di compressione e rarefazione

All'istante t_{-1}, ovvero prima dell'istante t_0 la pressione dell'aria è al suo valore normale, dato che la perturbazione non è ancora giunta al nostro punto di osservazione. All'istante t_0 la perturbazione giunge al nostro punto di osservazione, la pressione inizia a crescere, giunge al massimo all'istante t_1, poi decresce fino a tornare al valore normale all'istante t_2, continua a decrescere e giunge al minimo all'istante t_3, per poi risalire fino al valore normale all'istante t_4, e così via.
Si è fin qui descritto un **ciclo** del fenomeno. Se questo si ripete sempre allo stesso modo il fenomeno si dice periodico.[8] Il tempo necessario al completamento di un ciclo si dice **periodo**, si indica con il simbolo T e si misura in secondi (s) o in millisecondi (ms). L'inverso del periodo, cioè il numero di cicli che vengono completati in un secondo, si dice frequenza, e si misura in Hertz (Hz) o cicli per secondo (cps).
Se per esempio un'onda sonora ha periodo T=0.01 s (cioè 1/100 di secondo) la sua frequenza sarà di: 1/T = 1/0.01 = 100 Hz (o 100 cicli al secondo)" (ibidem).

Osservando la figura 1.9 ascoltiamo i suoni dell'esempio interattivo[9] numero 1A: possiamo constatare come, all'aumento del numero dei cicli al secondo (Hz), corrispondano suoni sempre più acuti.

[8] Matematicamente una forma d'onda si dice periodica se si ripete regolarmente e per un tempo infinito: nella pratica musicale, naturalmente, ci si "accontenta" di durate molto inferiori! In genere un'onda è "musicalmente periodica" quando, ripetendosi con regolarità, persiste per un tempo sufficiente a generare la sensazione di altezza corrispondente al periodo dell'onda. Approfondiremo la questione nel capitolo 2.

[9] Vi ricordiamo che gli esempi interattivi e gli altri materiali di supporto al libro si trovano all'indirizzo http://www.**********.

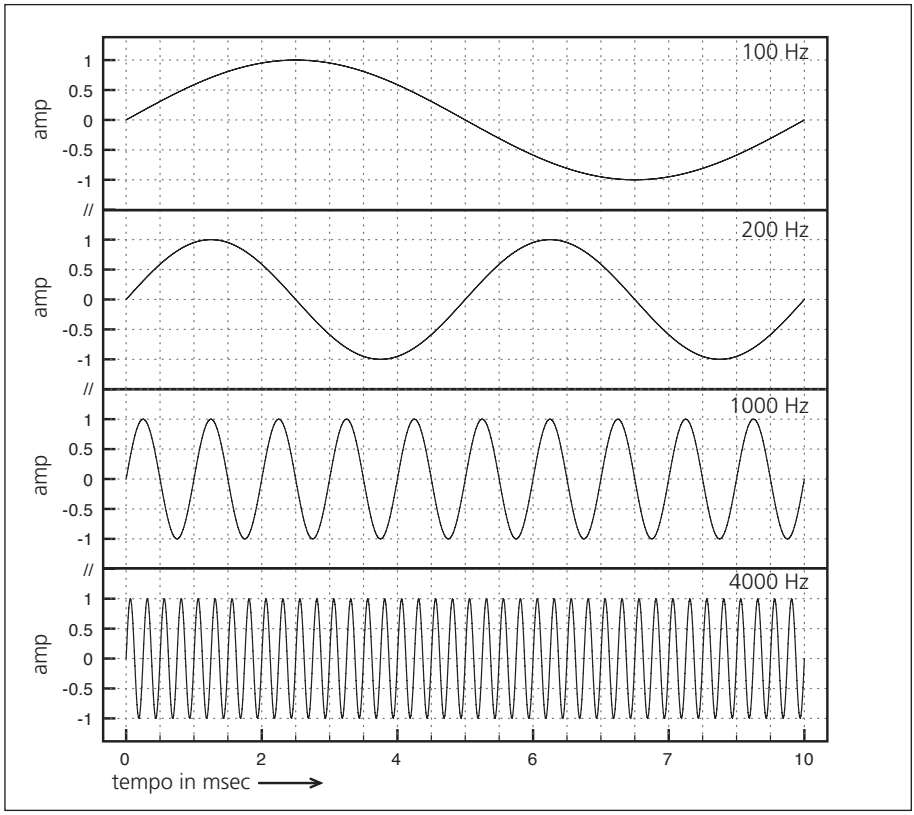


fig.1.9: quattro suoni di frequenza diversa

ESEMPIO INTERATTIVO 1A • FREQUENZA

Dal momento che si propaga nello spazio, un'onda ha una lunghezza che è inversamente proporzionale alla sua frequenza. Chiariamo questo concetto: la velocità del suono nell'aria (cioè la velocità con cui si propagano le onde sonore a partire dalla sorgente) è di circa 344 metri al secondo.[10] Questo significa che un'ipotetica onda di 1 Hz avrebbe una lunghezza di circa 344 metri, perché quando ha completato un ciclo è passato un secondo e in questo tempo si è dispiegata nello spazio per una lunghezza di 344 metri. Un'onda di 10 Hz, invece, in un secondo compie 10 cicli, che si dispongono nello spazio di 344 metri occupando ciascuno 34.4 metri, cioè un decimo dello spazio totale.

[10] Per la precisione questa velocità viene raggiunta quando la temperatura è di 21°. La velocità del suono, infatti, è proporzionale alla temperatura dell'aria.

Per lo stesso ragionamento un'onda di 100 Hz misura 3.44 metri: come si vede all'aumentare della frequenza diminuisce la lunghezza, e le due grandezze sono quindi, come abbiamo già detto, inversamente proporzionali.

AMPIEZZA

Il secondo parametro fondamentale del suono è l'**ampiezza**, che dà informazioni sulla variazione della pressione sonora, e che permette di distinguere un suono di forte intensità da uno di intensità debole.
La pressione sonora più debole che l'orecchio umano è in grado di percepire si dice **soglia inferiore di udibilità**, mentre la pressione sonora massima che un ascoltatore umano può sopportare si dice **soglia del dolore**, in quanto al di là di questa si ha una vera e propria sensazione di dolore fisico e danni permanenti all'organo dell'udito.
Osservando il fenomeno rappresentato in fig. 1.10, il valore massimo della pressione si dice **ampiezza di picco** dell'onda sonora; il valore della pressione in un punto qualsiasi si dice invece **ampiezza istantanea**.
Quando si indica l'ampiezza di un suono, ci si riferisce al valore dell'ampiezza di picco del suono stesso (vedi fig. 1.10).

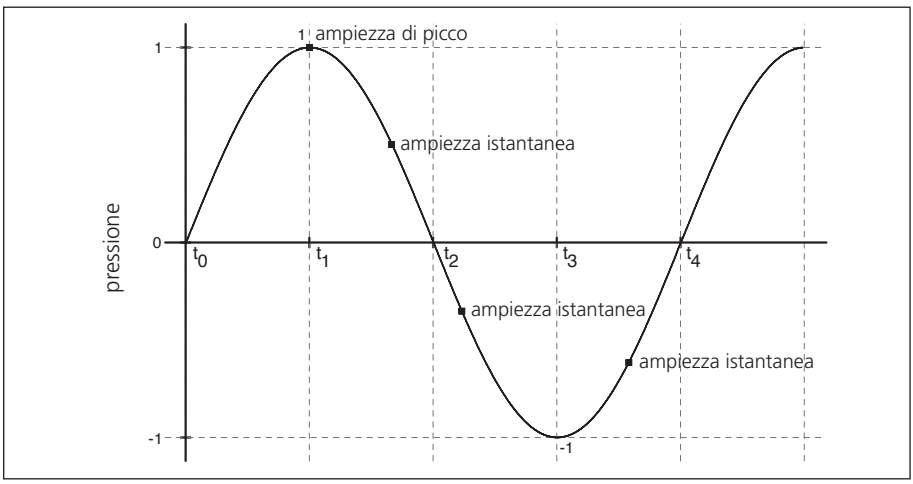


fig.1.10: ampiezza di un suono

Ad esempio, se indichiamo un'ampiezza di picco 1, avremo un'onda che parte da un'ampiezza istantanea 0 (all'istante t_0); poi l'ampiezza istantanea inizia a crescere, giunge al massimo all'istante t_1 (valore 1) poi decresce fino a tornare al valore 0 all'istante t_2, continua a decrescere e giunge al minimo all'istante t_3 (-1) per poi risalire fino al valore 0 all'istante t_4, e cosí via. Questa è la rappresentazione dell'ampiezza di un'onda sonora in funzione del tempo. Il processo di digitalizzazione trasforma tale ampiezza in una serie di numeri compresi tra 1 e -1.
I numeri così ottenuti possono essere usati per rappresentare graficamente la forma dell'onda (fig. 1.11). La posizione in cui si trova il ciclo di un'onda in un determinato istante viene chiamata **fase**.
Approfondiremo il concetto di fase nel par. 2.1.

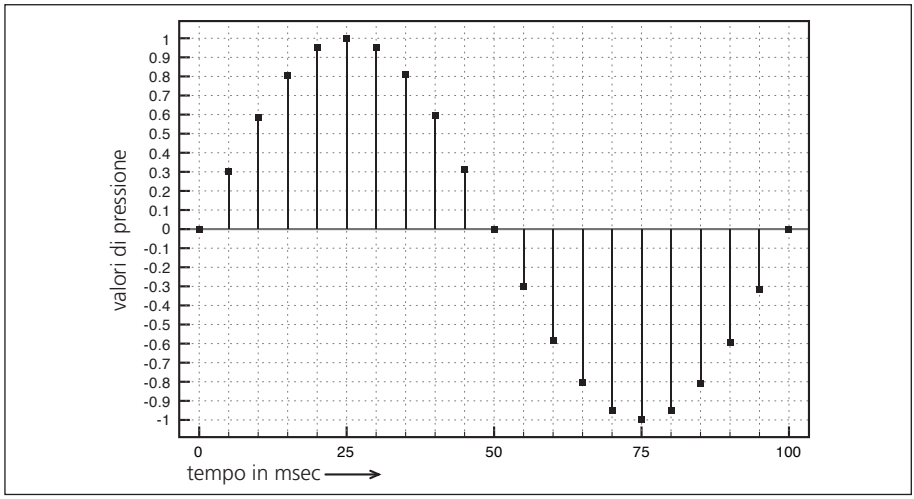


fig. 1.11: rappresentazione digitale di una forma d'onda.

Facendo un confronto con l'onda reale (cioè con la successione di compressioni ed espansioni delle molecole dell'aria), possiamo vedere che la compressione corrisponde ai numeri positivi e l'espansione ai numeri negativi, mentre il valore 0 indica una pressione non perturbata (l'assenza di segnale corrisponde digitalmente a una sequenza di zeri). I valori di ampiezza sono convenzionalmente espressi in numeri con la virgola e variano tra 0 e 1: se indichiamo 1 (cioè il valore massimo) come valore d'ampiezza di picco, avremo...

(...)

Il capitolo prosegue con:

Forma d'onda
La sinusoide
Altre forme d'onda
Onde bipolari e unipolari
Uso dei logaritmi nel calcolo dei decibel

1.3 VARIAZIONI DI FREQUENZA E AMPIEZZA NEL TEMPO: INVILUPPI E GLISSANDI
Inviluppi di strumenti acustici
Inviluppi di suoni sintetici
Glissandi
Curve esponenziali e logaritmiche

1.4 RAPPORTO TRA FREQUENZA E INTERVALLO MUSICALE

1.5 CENNI SULLA GESTIONE DEI SUONI CAMPIONATI
La digitalizzazione del suono

1.6 CENNI SUL PANNING

ATTIVITÀ
- Esempi interattivi

VERIFICHE
- Test a risposte brevi
- Test con ascolto e analisi

SUSSIDI DIDATTICI
- Concetti di base - Glossario - Un po' di storia

1P
SINTESI DEL SUONO CON MAX

CONTRATTO FORMATIVO

PREREQUISITI PER IL CAPITOLO

- Conoscenze di base degli strumenti informatici (operazioni base, gestione delle cartelle, scheda audio, etc.)
- Conoscenza minima della teoria musicale (toni, semitoni, ottave, tempi etc.)
- Contenuti del Cap. 1 della parte di teoria (si consiglia di studiare un capitolo per volta, affrontando prima la teoria e poi la pratica con Max)

OBIETTIVI

Abilità

- Saper utilizzare tutte le funzioni di base del software Max
- Saper sintetizzare suoni in sequenza e in sovrapposizione utilizzando oscillatori sinusoidali, ad onda quadra, triangolare o dente di sega
- Saper controllare in modo continuo l'ampiezza, la frequenza e la spazializzazione stereofonica di un suono (uso di spezzate di retta e di esponenziale per glissandi, inviluppi d'ampiezza e movimento del suono nello spazio stereo)
- Saper generare sequenze casuali di suoni sintetizzati
- Saper gestire l'utilizzo elementare dei suoni campionati

Competenze

- Saper realizzare un primo studio sonoro di due minuti basato sulle tecniche acquisite e memorizzarlo su file audio

CONTENUTI

- Sintesi ed elaborazione del suono al computer
- Timbro, altezza e intensità di un suono
- Glissando e inviluppo d'ampiezza
- Rapporti tra frequenze, altezze e codifiche MIDI
- Uso di suoni campionati (cenni)

TEMPI - Cap. 1 (Teoria e Pratica) + Interludio A

Autodidatti

Per 300 ore globali di studio individuale (vol. I, teoria e pratica): ca. 100 ore

Corsi

Per un corso globale di 60 ore in classe + 120 di studio individuale (vol. I, teoria e pratica): ca. 16 ore frontali + 4 di feedback - ca. 40 di studio individuale

ATTIVITÀ

- sostituzione di parti di algoritmi, correzione, completamento e analisi di algoritmi, costruzione di nuovi algoritmi

VERIFICHE

- Compiti unitari di Reverse Engineering

SUSSIDI DIDATTICI

- Lista comandi principali Max - Lista oggetti Max - Lista messaggi, attributi e parametri per oggetti Max specifici - Glossario

1.1 PRIMI PASSI CON MAX

Per procedere nella lettura di questo capitolo è necessario aver installato correttamente Max nel proprio computer. Se non l'avete fatto, potete scaricare l'applicazione nel sito http://www.cycling.74.com. Sarà inoltre necessario scaricare il materiale di supporto che trovate nel sito http://www.********** ********. Vi raccomandiamo in particolare di installare la libreria *Virtual Sound Macros*: seguite la procedura di installazione che trovate allo stesso indirizzo.

Lanciamo il programma Max e selezioniamo dal menù File la voce *New Patcher* (oppure digitiamo <Mac: Command–n> <Win: Control–n>)[1]: apparirà una finestra, la **Patcher Window** nella quale possiamo cominciare ad assemblare il nostro primo algoritmo Max. Prima di procedere, notate che la *Patcher Window* presenta una cornice scura contenente diverse icone (vedi fig. 1.1). I quattro lati della cornice sono denominati *Toolbar* e spiegheremo la funzione delle varie icone via via che ci serviranno.

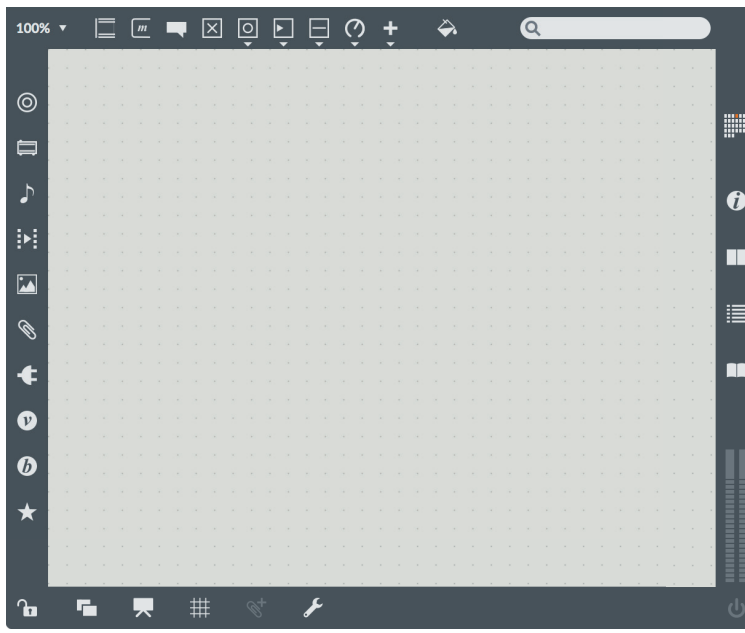


fig.1.1: La *Patcher Window*

Cominciamo subito con la *Toolbar* superiore: le icone che contiene rappresentano gli oggetti Max che abbiamo a disposizione per costruire una "macchina virtuale", o algoritmo di sintesi e/o elaborazione del suono. Gli oggetti Max sono infatti collegabili tra loro. Il flusso di informazioni (dati, numeri, segnali digitali...) passa da un oggetto all'altro attraverso questi collegamenti.

[1] Ovvero con Mac OS X teniamo pigiato il tasto Command (⌘) e digitiamo "n" e con Windows teniamo pigiato il tasto Control e digitiamo "n".

Ogni oggetto esegue un'operazione specifica sulle informazioni che riceve, e passa il risultato dell'elaborazione agli oggetti a cui è collegato.
Un insieme di oggetti collegati che svolge una determinata funzione si chiama **patch** (con riferimento ai sintetizzatori analogici modulari che vengono programmati con connessioni fisiche effettuate tramite cavi chiamati **patch cords**).
Cominciamo ora a realizzare la nostra prima *patch*. Se fate clic sulla prima icona a sinistra della *Toolbar* superiore, apparirà un oggetto grafico all'interno della *Patcher Window* (vedi fig. 1.2).

fig.1.2: l'*object box*

Questo è l'oggetto generico di Max e si chiama **object box**: è l'oggetto che useremo più spesso e la funzione che svolge dipende dal nome che gli diamo, cioè dalla stringa[2] che scriviamo al suo interno.
Vediamo innanzitutto come si crea un oscillatore sinusoidale. Proviamo a scrivere la parola "cycle~" all'interno dell'*object box*. Notate che, non appena cominciamo a scrivere, appare un menù che elenca tutti gli oggetti il cui nome o la cui descrizione contiene i caratteri che abbiamo digitato: questa utilissima funzione si chiama **autocompletion** (vedi figura 1.3).

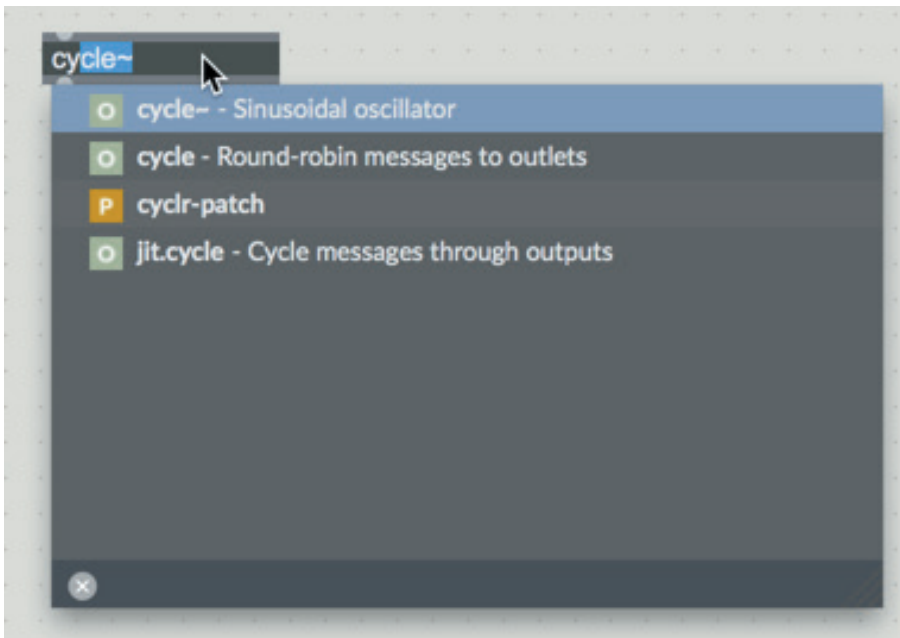


fig.1.3: il menù di *autocompletion*

[2] Per "stringa" intendiamo una sequenza di caratteri alfabetici e numerici: ad esempio "print", "salve" e "comma22" sono tutte stringhe.

In questa figura vediamo come appare il menù di *autocompletion* dopo che abbiamo digitato all'interno dell'*object box* i caratteri "cy" di "cycle~". La lettera che precede le voci del menù di *autocompletion* specifica il tipo di elemento individuato; nel menù in figura abbiamo due lettere: "o" indica un oggetto e "p" un'*abstraction*. Parleremo più avanti delle *abstraction* e dei vari tipi di elementi che troviamo in una *patch* Max. Notate che all'interno dell'oggetto appare, via via che inseriamo nuovi caratteri, il nome completo più probabile (in genere un nome che avevamo scelto in precedenza).
Da questo menù possiamo selezionare con un clic la voce che ci interessa: fate attenzione a selezionare la parola "cycle~" e non "cycle"![3]
Dopo aver selezionato "cycle~" digitate uno spazio; il menù di *autocompletion* ora mostrerà due nuove categorie "Arguments" e "Attributes": senza entrare nei dettagli, diciamo che gli elementi del menù sono ora dei "promemoria" di ciò che possiamo scrivere dopo il nome dell'oggetto. Ignoriamo questi promemoria per il momento e aggiungiamo dopo lo spazio (importantissimo!) il numero 440 all'interno dell'*object box*, dopo di che facciamo clic in un punto vuoto della *Patcher Window*.[4] L'*object box* dovrebbe assumere l'aspetto di figura 1.4.

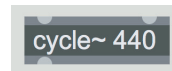


fig.1.4: l'oggetto `cycle~`

Le zone chiare nella parte alta e bassa dell'oggetto sono rispettivamente gli ingressi (**inlet**) e l'uscita (**outlet**), e vedremo tra poco come si utilizzano. (NB: Se l'oggetto non dovesse avere questo aspetto vuol dire che c'è un problema, leggetevi le FAQ alla fine di questo paragrafo).

[3] Notate il carattere che segue la parola cycle, "~", che si chiama **tilde** e che serve a contraddistinguere gli oggetti che elaborano il segnale digitale. Alcuni oggetti esterni potrebbero non comparire nel menù di *autocompletion* e sarà quindi necessario digitarli direttamente all'interno dell'*object box*: in questo caso è indispensabile sapere come creare una tilde. Questo carattere infatti si ottiene con una combinazione di tasti che varia a seconda del sistema operativo utilizzato e della nazionalità del layout di tastiera. Ad esempio sulla tastiera italiana del Macintosh si realizza con alt-5. Sulla gran parte dei PC Windows si scrive alt-126, usando la tastiera numerica a destra, altrimenti, se è assente, come nei portatili, si può tenere premuto il tasto fn per attivare la tastiera numerica interna ai tasti delle lettere e digitare alt-126. Se non funziona si può sfruttare l'*autocompletion* digitando il nome di un oggetto qualsiasi munito di tilde (come ad esempio `cycle~`) e sostituire manualmente il nome nell'*object box* (ovviamente conservando la tilde!).

[4] O in alternativa premiamo Enter su Macintosh o Maiuscole-Enter su Windows.

Ora creiamo un altro oggetto, **`gain~`**, che ha l'aspetto del *fader* di un mixer (vedi fig. 1.5). È sufficiente fare clic sulla settima icona della *Toolbar* superiore: apparirà una *palette* (tavolozza) contenente tutti gli oggetti della categoria "Slider". Facciamo clic sulla terza icona del menù, oppure trasciniamo l'icona sulla *Patcher Window* con il mouse.

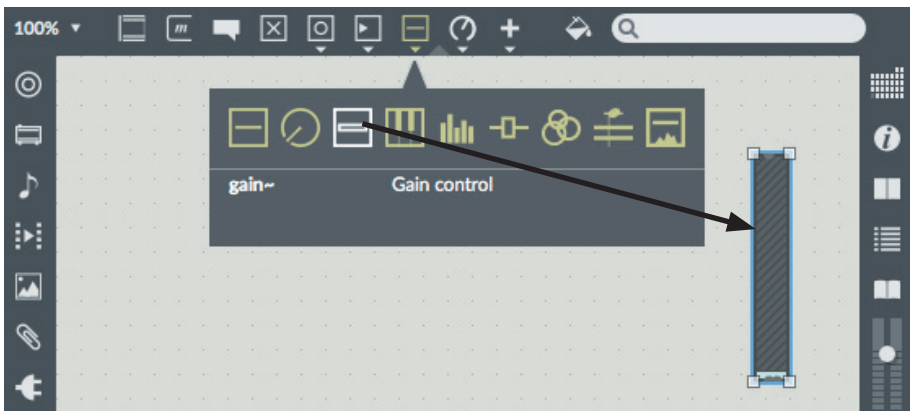


fig.1.5: l'oggetto `gain~`

In questo caso non si tratta di un *object box*, ma di un oggetto grafico, ovvero uno ***user interface object*** (***ui object***), un oggetto per l'interfaccia utente. Piccolo trucco: se non riuscite a trovare un oggetto nella *Toolbar* superiore prendete un *object box* generico (come il primo che abbiamo usato), scriveteci dentro il nome dell'oggetto desiderato, ad esempio `gain~`, fate clic all'esterno dell'oggetto e questo si trasformerà nel relativo *ui object*.
Spostate questo oggetto sotto `cycle~`, e collegate l'uscita di `cycle~` con l'ingresso di `gain~` in questo modo: avvicinate il puntatore del mouse all'uscita che si trova sotto l'oggetto `cycle~` e quando appare un cerchio giallo e un riquadro (detto *Assistance Bubble*) che indica la funzione dell'uscita selezionata (vedi fig. 1.6a) fate clic con il mouse e spostatelo verso il basso (apparirà un cavo giallo e nero). Quando il puntatore del mouse si avvicina all'angolo in alto a sinistra dell'oggetto `gain~`, apparirà un cerchio rosso con un *Assistance Bubble* che indica la funzione dell'ingresso di `gain~` (vedi fig. 1.6b); a quel punto fate nuovamente clic con il mouse: il collegamento tra i due oggetti è effettuato.

a

b

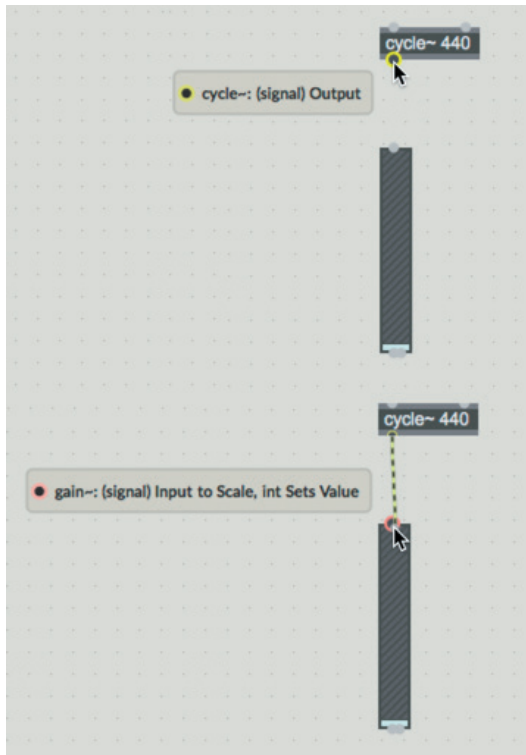


fig. 1.6: collegare gli oggetti

Abbiamo collegato l'oscillatore all'oggetto **`gain~`** che serve a regolarne il volume. Adesso dobbiamo inviare il segnale audio all'uscita; fate clic sulla penultima icona della *Toolbar* superiore, apparirà una *palette* con sei categorie di oggetti[5]: selezionate la categoria "Audio" e fate clic sulla seconda icona (oppure trascinatela nella *Patcher Window*: vedi fig. 1.7). Il nome di questo oggetto è **`ezdac~`**.

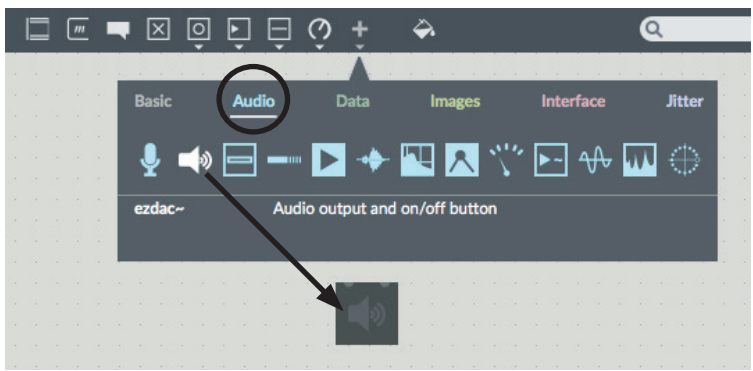


fig.1.7: l'oggetto **`ezdac~`**

[5] Tra poco, in questo stesso paragrafo, vedremo meglio come è organizzata la *Toolbar* superiore.

Spostiamolo sotto l'oggetto **gain~** e colleghiamo l'uscita di sinistra di quest'ultimo con i due ingressi di **ezdac~** (vedi fig. 1.8).

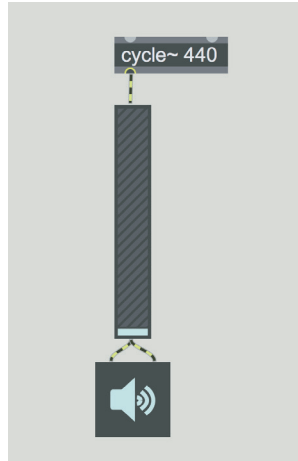


fig.1.8: la nostra prima *patch*

Attenzione! L'oggetto **gain~** ha due uscite scarsamente distinguibili: verificate quindi di aver usato l'uscita sinistra di **gain~** per entrambi i collegamenti. Il modo migliore per assicurarsi di aver usato l'uscita giusta è leggere l'*Assistance Bubble* che appare alla base di **gain~** quando effettuiamo il collegamento, e che deve contenere questo testo: "**gain~: (signal) Scaled Output**".
Se uno dei due cavi dovesse essere grigio, e non giallo-nero come appare nella figura qui sopra, significa che avete usato per sbaglio l'uscita di destra, e dovrete quindi cancellare il cavo in questo modo: selezionatelo con un clic (il cavo apparirà "ingrossato") e premete il tasto di cancellazione (quello che usate quando dovete cancellare del testo), a questo punto ricollegate gli oggetti nel modo corretto.
Probabilmente ora vorrete salvare la *patch* su disco; fatelo pure, ma con un'avvertenza: NON date alla *patch* lo stesso nome di un oggetto Max! Ad esempio, non chiamate questa *patch* "cycle~" (e nemmeno "cycle", senza tilde), è il modo migliore per confondere Max e avere risultati inaspettati la prossima volta che ricaricherete la *patch*. Dal momento che è impossibile ricordare tutti i nomi degli oggetti Max (per evitare di usarli come nome di *patch*), un buon modo per scongiurare il "pericolo" è dare al file un nome composto da più parole, ad esempio "test oscillatore", oppure "test oggetto cycle~", o quello che preferite: nessun oggetto Max ha un nome composto da più parole. Non trascurate questo consiglio, molti dei malfunzionamenti riscontrati dagli utenti Max alle prime armi derivano proprio dal fatto che prima o poi creano un file con lo stesso nome di un oggetto. Torneremo sull'argomento nell'"Interludio" che segue questo capitolo.
Bene, abbiamo realizzato la nostra prima *patch* e siamo pronti per farla funzionare. Manca però ancora un passaggio: finora abbiamo lavorato in **edit mode** cioè la modalità che ci permette di assemblare la *patch* spostando e collegando gli oggetti; ora, per far suonare la nostra *patch* dobbiamo passare

in **performance mode**, facendo clic sul piccolo lucchetto che appare in basso a sinistra nella *Patcher Window*, oppure premendo <Mac: Command–e> <Win: Control–e>.[6] Quando siamo in modalità *performance* il lucchetto in basso a sinistra appare chiuso (se lo vedete aperto vuol dire che siete in modalità *edit*!).
Adesso facciamo clic sull'oggetto **`ezdac~`** (il piccolo altoparlante), l'icona si "illuminerà", poi alziamo lentamente il cursore di **`gain~`**, dovremmo udire un suono, per la precisione un La sopra il Do centrale. Facendo nuovamente clic sul piccolo altoparlante possiamo "spegnere" la *patch*. Se non avete sentito alcun suono consultate le FAQ alla fine di questo paragrafo.
Analizziamo ora il nostro algoritmo: l'oggetto **`cycle~`** è un oscillatore, ovvero un generatore di suono che nel nostro caso genera un'onda sinusoidale, e il numero 440 indica la sua frequenza; questa sinusoide[7] cioè si ripete 440 volte al secondo.[8]
In altre parole **`cycle~`** è il nome dell'oggetto e 440 è il suo **argomento**, vale a dire il valore che l'oggetto in questione utilizza per operare, in questo caso appunto 440 Hz.
Questo oggetto è collegato con l'oggetto **`gain~`** e quindi il segnale che genera viene passato a quest'ultimo, che come abbiamo visto modifica il volume del segnale. Il segnale modificato passa poi ad **`ezdac~`** (il piccolo altoparlante), la cui funzione è quella di mandare il segnale alla scheda audio del computer. Quest'ultima effettua la conversione digitale-analogica del segnale, cioè trasforma i numeri in segnali audio che possiamo udire attraverso le casse collegate al computer. Il nome "ezdac" peraltro è un quasi-acronimo che sta per EaSy Digital to Analog Converter (Semplice Convertitore Digitale-Analogico).
Cerchiamo di approfondire ulteriormente questa *patch*; oltre ad udire il suono, infatti, possiamo "vederlo". Salviamo la *patch* che abbiamo appena realizzato in una cartella apposita che potreste chiamare, ad esempio, "le mie patch" (ci servirà nel prossimo paragrafo) e chiudiamo la *Patcher Window*.
Ora scaricate (se non l'avete ancora fatto) il "Materiale Capitoli Max Vol 1" che si trova all'indirizzo www.**********. Poi aprite il file **01_01.maxpat** che trovate nella cartella "Materiale Capitoli Max Vol 1/Patch Max Vol 1/Capitolo 01 Patch".

[6] In alternativa è possibile passare alla modalità performance tenendo premuto il tasto <Mac: Command> <Win: Control> facendo clic con il tasto sinistro del mouse su un'area vuota della *Patcher Window*.
[7] In realtà, come vedremo nel prossimo capitolo, si tratta di un coseno
[8] Tutti questi concetti vengono spiegati nel paragrafo 1.2 della parte di teoria.

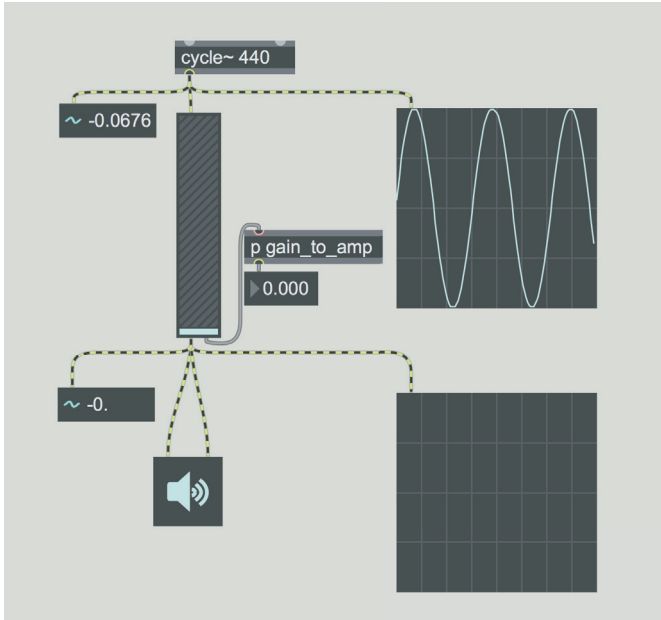


fig.1.9: file 01_01.maxpat

Qui abbiamo aggiunto alla *patch* alcuni nuovi oggetti. Gli oggetti sulla sinistra in cui sono visibili dei valori numerici si chiamano **`number~`** e mostrano, sotto forma di numero, il contenuto del segnale che ricevono; gli oggetti quadrati sulla destra si chiamano **`scope~`**[9] e sono degli oscilloscopi che ci fanno vedere il segnale come un'onda che si muove su uno schermo; l'oggetto [`p` gain_to_amp] e l'oggetto collegato (che si chiama **`flonum`** o *float number box*) ci fanno vedere di quanto `gain~` amplifica o attenua il segnale che riceve.
Avviamo l'algoritmo facendo clic sull'oggetto `ezdac~` e osserviamo i numeri mostrati dal `number~` in alto a sinistra: questi numeri sono prodotti dall'oggetto `cycle~` e, se li osserviamo per un po' ci renderemo conto che sono valori, positivi e negativi, compresi tra 1 e -1. Sul lato destro vediamo lo `scope~` superiore che ci mostra questi stessi numeri sotto forma di grafico: nella metà superiore del riquadro vengono rappresentati i valori positivi, in quella inferiore i negativi. Nel riquadro dello `scope~` viene mostrato non un singolo numero, ma una sequenza di diverse centinaia di elementi, che vengono visualizzati come punti nel riquadro stesso: questi punti sono molto vicini tra loro e nell'insieme ci appaiono come una linea curva. Questi elementi, questi numeri, nella terminologia della musica digitale si chiamano *campioni*. La linea che oscilla sinuosamente in alto e in basso all'interno dell'oscilloscopio è appunto la forma d'onda sinusoidale prodotta da `cycle~`.

[9] Gli oggetti `number~` e `scope~` si trovano come i precedenti nella *Toolbar* superiore: fate clic sulla penultima icona e selezionate la categoria "Audio". Se non riuscite a trovarli, potete usare il trucco che vi abbiamo spiegato sopra: prendete un *object box* e ci scrivete dentro il nome dell'oggetto grafico desiderato.

Ci sono un altro `number~` e un altro `scope~` collegati all'oggetto `gain~`, che ci mostrano rispettivamente il numero 0 e una linea piatta (che è una sequenza di zeri). Questo perché il cursore è abbassato, cioè il volume è a zero.
Se alziamo il cursore di `gain~` vediamo che il `number~` ci mostra dei numeri dapprima molto piccoli e poi via via sempre più grandi man mano che aumentiamo il volume. Nello stesso tempo la linea piatta dello `scope~` in basso comincia a diventare ondulata e ad assomigliare a quella dello `scope~` in alto: quella che viene modificata è cioè l'*ampiezza* del segnale; più alziamo il cursore e più l'oscillazione diventa ampia.
Se però alziamo troppo il cursore di `gain~` vediamo che i numeri cominciano a superare i limiti di 1 e -1, che la forma d'onda, rappresentata nell'oscilloscopio, diventata troppo ampia e appare tagliata, e soprattutto che il suono cambia, diventa distorto.

Da tutto ciò possiamo trarre alcune conclusioni:
1) L'oggetto `cycle~` produce una sequenza di valori digitali che seguono l'andamento di una (co)sinusoide.
2) I limiti numerici di questa sinusoide sono 1 e –1. Come si vede nell'immagine che appare nello `scope~` superiore, questi sono anche i limiti massimi, superati i quali il suono viene distorto.
3) L'oggetto `gain~` modifica l'ampiezza della sinusoide, e fa sì che i campioni in entrata siano diversi dai campioni in uscita. Come fa? *Moltiplicando* i valori che riceve per una certa quantità che dipende dalla posizione del cursore. Quando il cursore è nella posizione più bassa il segnale viene moltiplicato per 0, e il risultato è una sequenza di zeri, come abbiamo visto, perché qualsiasi numero moltiplicato per 0 dà come risultato 0. Man mano che alziamo il cursore il fattore di moltiplicazione aumenta.
Se ad esempio lo portiamo a 0.5 l'ampiezza dei campioni che entrano nel `gain~` viene dimezzata (perché moltiplicare un numero per 0.5 equivale a dividerlo per 2).[10] Se poi lo portiamo ad 1 (spostando il cursore a circa 3/4 dell'altezza del fader) i campioni in entrata non subiscono variazioni in uscita, rimangono identici.
Infine alziamo ulteriormente il cursore. Ora i valori estremi dei campioni superano il limite di 1 e -1, ma questi campioni vengono riportati entro i limiti durante la conversione digitale-analogica: questo fa sì che la forma d'onda non sia più una sinusoide, poiché l'onda appare tagliata (come vediamo nell'oscilloscopio inferiore). In realtà i campioni fuori range vengono semplicemente riportati alla massima ampiezza disponibile, e il suono distorto che sentiamo è relativo a questa nuova forma d'onda.

[10] Per portare il cursore ad una altezza che corrisponda ad una moltiplicazione per 0.5 controllare che il *number box* collegato all'oggetto [`p` gain_to_amp] mostri il valore 0.5. In realtà l'incremento del *fader* è logaritmico, secondo una formula che non è il caso di spiegare qui, e l'oggetto [`p` gain_to_amp] serve appunto a convertire la posizione del *fader* (che viene prodotta all'uscita di destra di `gain~`) nell'effettiva ampiezza. Non vediamo in dettaglio come funziona questo oggetto perché non abbiamo ancora le conoscenze sufficienti a capirlo: approfondiremo la questione nell'interludio A che segue questo capitolo. Notate comunque che quando il fattore di moltiplicazione è all'incirca 0.5 la sinusoide occupa metà del riquadro.

Abbiamo trattato più a fondo i concetti di ampiezza, frequenza e forma d'onda nel par. 1.2 della teoria, riassumiamo alcuni concetti basilari:

- l'*ampiezza* è il parametro fisico da cui dipende l'*intensità* del suono, cioè il parametro che ci fa percepire forte o piano un determinato evento sonoro; i valori assoluti d'ampiezza (cioè indipendenti dal segno) in Max vanno da un minimo di 0 a un massimo di 1;
- la *frequenza* è il parametro fisico da cui dipende l'altezza del suono, cioè il parametro che ci fa percepire un suono come grave o acuto. I valori sono espressi in Hertz (Hz), e quindi dovremo tener conto che i suoni udibili dall'uomo sono fra circa 20 e circa 20000 Hz;
- la *forma d'onda*, che nel caso di `cycle~` come abbiamo visto è una sinusoide, è un parametro fondamentale che concorre a definire il *timbro* del suono, cioè quella qualità del suono che ci consente di percepire la differenza, ad esempio, fra il Do di una chitarra e quello di un sassofono.

FAQ (Frequently Asked Questions)

FAQ significa "Domande Frequenti" e in questa sezione cercheremo di dare una risposta ad alcuni dei problemi più comuni che si incontrano quando si comincia a lavorare con Max. Leggetele attentamente anche se non avete incontrato alcun problema, contengono informazioni che vi saranno utili nel seguito della lettura di questo libro.

1) Domanda: Ho creato un oggetto chiamato "cycle~440" come c'è scritto in questo capitolo, ma l'oggetto non ha né ingressi né uscite. Perché?

Risposta: Controllate di aver messo uno spazio tra "cycle~" e "440" perché il primo è il nome dell'oggetto e il secondo è l'argomento, che in questo caso rappresenta la frequenza del suono. Se le due parole sono attaccate Max cercherà un oggetto inesistente chiamato "cycle~440" e non trovandolo non mostrerà un *object box* corretto con ingressi e uscite.

2) D: Va bene. Perché però non mi ha dato un messaggio di errore?

R: Il messaggio d'errore c'è, e si trova nella ***Max Console***: una finestra che il programma utilizza per comunicare con l'utente. Per richiamarla digitate <Mac: Command–m> <Win: Control–m>, oppure fate clic sulla quarta icona della *Toolbar* di destra: apparirà una finestra alla destra della *Patcher Window*. Nella finestra, troverete probabilmente questo messaggio:
"cycle~440: No such object"
Se fate doppio clic sul messaggio di errore, l'oggetto che lo ha generato (in questo caso l'inesistente "cycle~440") verrà evidenziato nella *Patcher Window*.

3) D: Io ho messo uno spazio tra "`cycle~`" e "440", però l'oggetto è privo di ingressi e uscite lo stesso!

R: Questo è un errore più sottile, e capita spesso all'inizio con gli oggetti che hanno una tilde (~) alla fine del nome. Probabilmente per scrivere questo carattere avete dovuto usare una combinazione di tasti (ad esempio, per Mac <*alt–5*>), e uno dei tasti della combinazione è rimasto premuto mentre avete digitato lo spazio (avete ad esempio premuto <*alt–spazio*>); la combinazione non è riconosciuta da Max che quindi non è in grado di separare il nome dell'oggetto dall'argomento. Cancellate lo spazio e riscrivetelo, facendo attenzione a premere solo la barra spaziatrice.

4) D: Non sento alcun suono.

R: Avete fatto clic sull'oggetto **`ezdac~`** (il piccolo altoparlante)? Avete alzato il cursore del *fader*? Siete sicuri che il computer non sia in *mute*, ovvero riuscite a riprodurre dei suoni con altri programmi? Avete controllato che sulla finestra *Audio Stat*us (sotto il menù *Options*) sia stata selezionata la scheda audio giusta?

PICCOLO "MANUALE DI SOPRAVVIVENZA" PER MAX

In questa sezione daremo alcune informazioni essenziali per muoversi bene nell'ambiente Max.

COMANDI DA TASTIERA BASILARI
Innanzitutto rivediamo i comandi da tastiera che abbiamo imparato finora:

<Mac: Command–n> <Win: Control–n> serve a creare una nuova *Patcher Window*, il nostro spazio di lavoro dove possiamo realizzare le *patch*. .

<Mac: Command–e> <Win: Control–e> serve per alternare la modalità *edit* alla modalità *performance* nella *Patcher Window*. In *edit* possiamo assemblare le *patch* prendendo gli oggetti dalla *Toolbar* superiore; in *performance* possiamo far funzionare la *patch* ed interagire con gli oggetti grafici di interfaccia, come i *float number box* o l'oggetto **`gain~`**.

<Mac: Command–m> <Win: Control–m> serve per richiamare (qualora non fosse già visibile) la *Max Console* che è una finestra utilizzata dal programma per comunicare con l'utente, e che l'utente può usare per visualizzare brevi messaggi (vedremo più avanti come).

Inoltre è possibile creare degli oggetti digitando un semplice carattere, senza tasti modificatori come Command o Control: con "*n*" ad esempio possiamo creare (in modalità *edit*) un *object box* vuoto nella posizione del puntatore del mouse, esattamente come quello che otterremmo dalla prima icona della *Toolbar* superiore. Ci sono altri tasti che ci permettono di creare oggetti; in una *Patcher Window* vuota provate a digitare *"f" "i" "t" "b"* mentre spostate il puntatore del mouse: otterrete diversi oggetti (al momento per voi assolutamente sconosciuti!) che utilizzeremo molto spesso nel corso dei prossimi capitoli.

SELEZIONARE, CANCELLARE E COPIARE

Per cancellare un cavo o un oggetto bisogna assicurarsi di essere in modalità *edit*[11] e poi selezionarlo con il mouse e premere il tasto di cancellazione, detto anche *backspace*. Possiamo selezionare più oggetti contemporaneamente facendo clic su un punto vuoto della *Patcher Window* e trascinando il mouse in modo da includere gli oggetti da selezionare nell'area di trascinamento (vedi fig. 1.10).

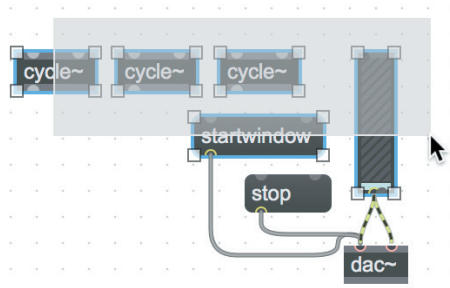


fig.1.10: selezionare gli oggetti

A questo punto se spostiamo uno degli oggetti selezionati spostiamo anche tutti gli altri, oppure se premiamo il tasto di cancellazione li cancelliamo tutti. Con questa procedura vengono selezionati gli oggetti ma non i cavi; se abbiamo bisogno di selezionare più cavi contemporaneamente (ad esempio per cancellarli) dobbiamo premere il tasto *Alt* mentre trasciniamo il mouse e "tocchiamo" i cavi che ci interessano (vedi fig. 1.11).

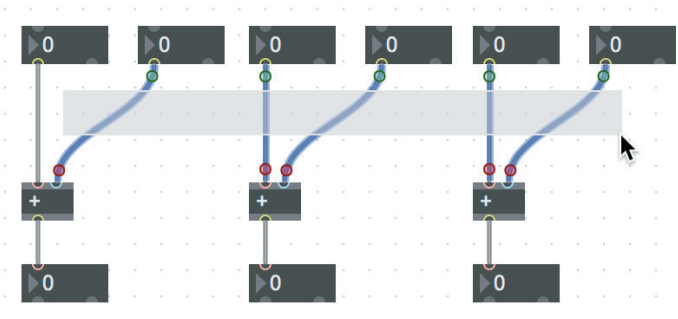


fig.1.11: selezionare i cavi

[11] Ovvero bisogna assicurarsi che il lucchetto che si trova in basso a sinistra nella finestra che contiene la *patch* sia aperto.

Sempre con il tasto *Alt* premuto potete duplicare un oggetto facendoci clic sopra e trascinandolo. Se prima selezionate più oggetti e poi ne trascinate uno con *Alt-clic*, li copierete tutti (vedi fig. 1.12).

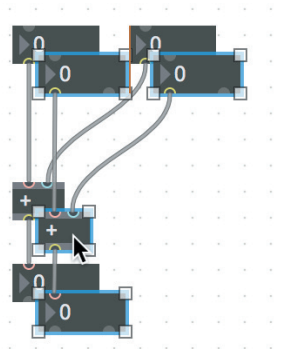


fig.1.12: copia di un insieme di oggetti

Se fate un errore (ad esempio cancellate un oggetto al posto di un altro) potete annullarlo selezionando il comando "*undo*" dal menù *Edit* (in italiano il comando si traduce generalmente con *annulla*). Se poi vi accorgete che non era un errore (ad esempio che volevate cancellare proprio quell'oggetto) potrete ripristinare la situazione tramite il comando "*redo*" (ripristina) sempre dal menù *Edit*. Selezionando ripetutamente il comando "*undo*" potete annullare una sequenza di azioni e riportare la *patch* ad uno stato precedente: da tastiera il comando equivalente a "*undo*" è <Mac: Command–z> <Win: Control–z>, per il "*redo*" è <Mac: Shift-Command–z> <Win: Shift- Control–z>.[12]

DOCUMENTAZIONE E HELP
Questo libro è autosufficiente: qui troverete tutte le informazioni utili per comprendere e usare le *patch* che via via illustreremo e per utilizzare le diverse tecniche di sintesi ed elaborazione del suono con Max. Se conoscete l'inglese può essere utile dare anche un'occhiata alla documentazione e al sistema di Help in linea del programma.

[12] Lo Shift è il tasto delle maiuscole.

Selezionando la voce *Reference* dal **menù Help** otteniamo la finestra di fig. 1.13 (che potrebbe essere differente per le diverse versioni di Max).

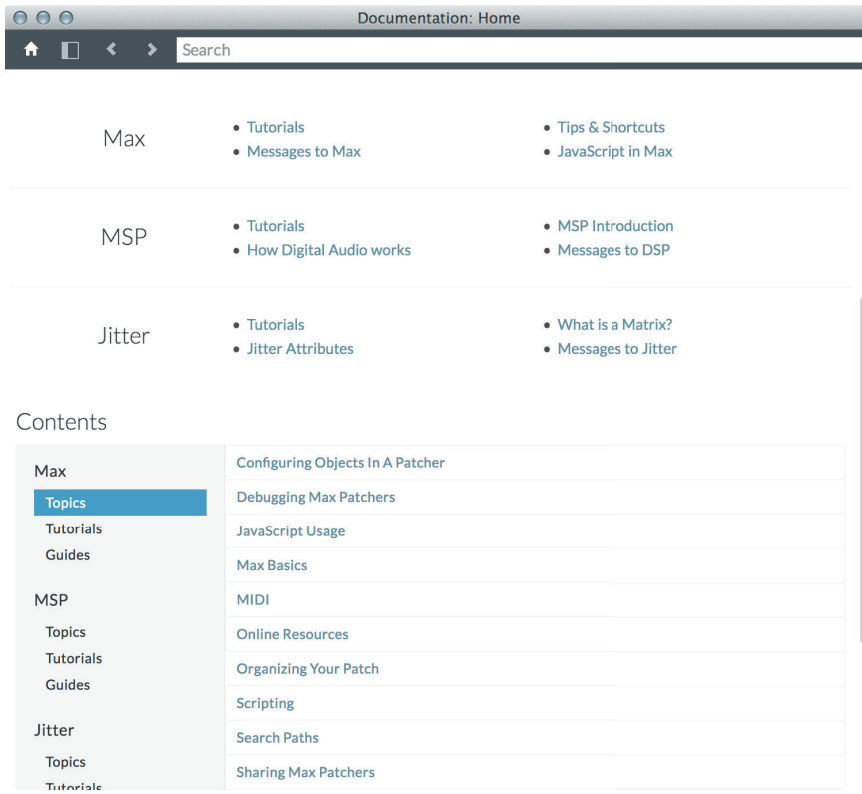


fig.1.13: la finestra *Documentation*

In figura vediamo una parte della finestra *Documentation* di Max. In alto ci sono le tre categorie principali: **Max** (il sistema che gestisce le funzioni di controllo, ovvero tutto ciò che non è generazione audio o video/grafica), **MSP** (generazione ed elaborazione di segnali audio) e **Jitter** (elaborazione video, grafica, gestione di matrici). Per ogni categoria ci sono dei Tutorial che introducono gradualmente i diversi elementi, più alcuni degli argomenti principali.
In basso c'è un indice dei contenuti che riporta per ogni categoria tutti gli argomenti (topics) inerenti, la lista dei tutorial, e una serie di guide che approfondiscono tutti gli aspetti della programmazione e della gestione dell'ambente Max. Provate a fare un "giro" in questa documentazione per capirne meglio il funzionamento.
Ci sono anche le *patch* di help dei singoli oggetti (sempre in inglese): se in modalità *edit* fate "*Alt-clic*" (senza trascinare) su un oggetto, si aprirà una *patch* di aiuto relativa all'oggetto; questa *patch* è perfettamente funzionante e riassume le caratteristiche principali dell'oggetto selezionato. Facendo "*Alt-clic*" in modalità *edit* sull'oggetto `cycle~` ad esempio si ottiene la **help patch** di fig. 1.14 (potrebbe essere differente per differenti versioni di Max).

fig.1.14: una *patch* di help

Le *patch* di help hanno una struttura particolare: sono suddivise in schede richiamabili tramite le etichette visibili nella parte alta della finestra. Ciascuna scheda spiega caratteristiche diverse dell'oggetto. Il numero e la denominazione delle etichette varia da oggetto a oggetto, ad esclusione della prima e dell'ultima che sono comuni a tutti gli oggetti. La prima etichetta ("basic") illustra le funzioni fondamentali dell'oggetto; l'ultima etichetta, recante un punto interrogativo, fa apparire un menù: selezionando la prima voce, "Open Reference", di questo menù, è possibile visualizzare una pagina del manuale di riferimento in cui vengono spiegate dettagliatamente tutte le caratteristiche dell'oggetto. Le voci successive richiamano le *patch* di help di oggetti che svolgono funzioni analoghe o sono spesso utilizzati insieme all'oggetto in questione. Le ultime voci richiamano una serie di tutorial che impiegano il nostro oggetto. Se conoscete l'inglese tecnico potrà esservi utile consultare gli help per scoprire o ricordare tutti i dettagli di cui avete bisogno.
Anche se non conoscete una parola di inglese, però, vi consigliamo lo stesso di dare un'occhiata alle *patch* di help; innanzitutto perché sono *patch* funzionanti e quindi si possono imparare molte cose utilizzandole, e poi perché molti termini, come "oscillator", "frequency" o "intensity" etc., non sono difficili da interpretare.[13]

[13] Vi ricordiamo inoltre che nei capitoli di teoria sono riportate le traduzioni dall'italiano all'inglese di molti termini tecnici.

Un'altra fonte di informazione è la **Clue Window**, richiamabile dal menù *Window*: questa finestra, che con le impostazioni di *default*[14] appare come un piccolo riquadro giallo, visualizza informazioni relative a ciò che si trova sotto il puntatore del mouse. Provate ad attivarla e a portare il puntatore del mouse sui vari elementi di una *patch* o sulle diverse voci dei menù: di ognuno di questi elementi la *Clue Window* vi mostrerà una breve descrizione.
Il sistema di aiuto alla programmazione è sicuramente uno dei punti di forza di Max: oltre agli help e alla *Clue Window*, abbiamo incontrato più sopra gli *Assistance Bubble* che ci danno informazioni sui messaggi che gli oggetti inviano o che possono ricevere. Ricordiamo che per visualizzare un *Assistance Bubble* è sufficiente, in modalità *edit*, portare il puntatore del mouse sopra un ingresso o un'uscita di un oggetto (vedi figg. 1.6a e 1.6b).
Vediamo adesso un'altra risorsa, il **Quickref Menu**: aprite nuovamente la patch *01_01.maxpat* e andate in modalità *edit* aprendo con un clic il piccolo lucchetto che si trova in basso a sinistra nella *Patcher Window*. Ora portate il puntatore del mouse al di sopra dell'ingresso di sinistra di `cycle~` in modo che compaia il cerchio rosso e l'*Assistance Bubble*, fate clic con il tasto destro del mouse all'interno del cerchio rosso e tenete pigiato il tasto: apparirà il menù *Quickref* (vedi fig. 1.15).

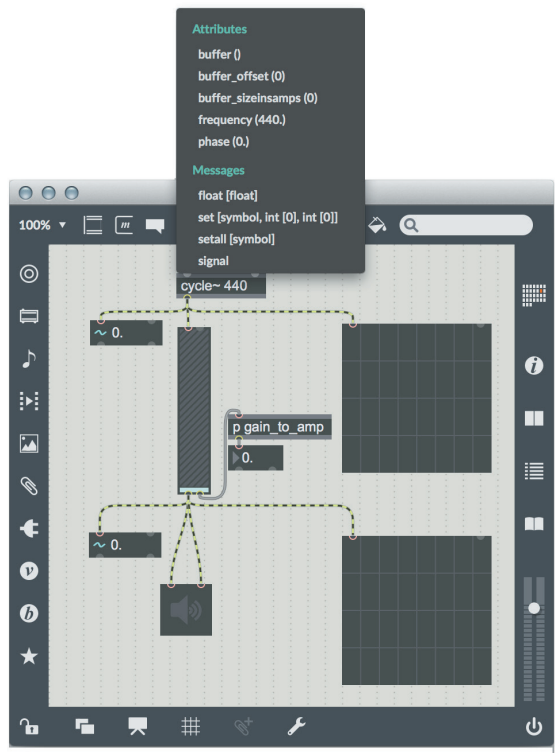


fig.1.15: *Quickref* Menu

[14] Per impostazioni di default si intendono le impostazioni "di fabbrica" che possono essere modificate dall'utente.

Questo menù contiene due categorie di elementi, gli **Attributes** (Attributi) di cui parleremo più avanti e i **Messages** (Messaggi) che corrispondono ai tipi di dati che l'oggetto è in grado di "comprendere" e utilizzare: selezionando uno di questi messaggi è possibile creare un oggetto che si collega "automaticamente" a `cycle~`. Fate ad esempio clic sulla voce *float [float],* apparirà un nuovo oggetto connesso a `cycle~` (vedi fig. 1.16)

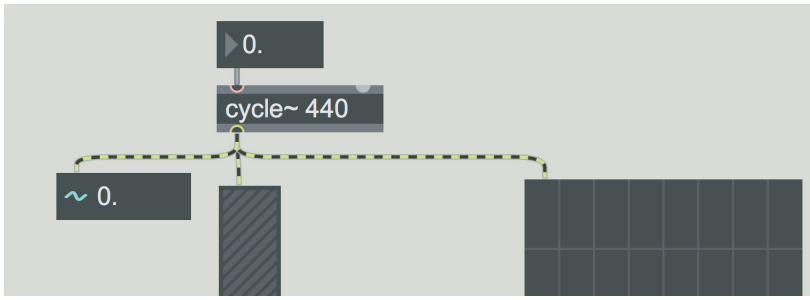


fig. 1.16: collegare un oggetto tramite *Quickref*

Se ora andate in modalità *performance* (chiudendo con un clic il piccolo lucchetto in basso a sinistra) e fate scorrere verticalmente il mouse con il tasto premuto sul nuovo oggetto modificherete il numero contenuto al suo interno. Questo oggetto infatti gestisce i valori numerici e si chiama **number** o *number box*: i numeri che abbiamo generato facendo scorrere il mouse sono stati inviati a `cycle~` e ne hanno modificato la frequenza (notate però che l'argomento 440 che si trova all'interno dell'oggetto `cycle~` non cambia, ma viene comunque annullato dai nuovi valori trasmessi). Provate ad avviare la *patch* facendo clic sull'icona dell'altoparlante, e alzate il cursore del fader fino a circa tre quarti, ora fate scorrere i numeri del *number box* su valori compresi tra 500 e 1000: sentirete l'oscillatore sinusoidale suonare alle diverse frequenze mostrate dal *number box*. Parleremo più diffusamente dell'oggetto *number box* nel seguito di questo capitolo.
Se ora portate il puntatore del mouse sul lato sinistro di un oggetto (sempre in modalità *edit*) vedrete comparire un cerchio giallo con un triangolo all'interno (fig. 1.16b), fate clic su questa icona e comparirà un nuovo menù contestuale, si tratta dell'*Object Action Menu*.

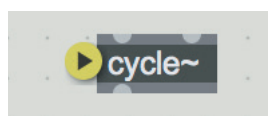


fig. 1.16b: icona *Object Action*

Questo menu (fig. 1.16c) contiene tutti gli elementi utili per gestire l'oggetto, trasformarlo, e visualizzare informazioni sul suo utilizzo. In pratica tutti gli elementi che trovate qui sono raggiungibili anche in altri modi (vedremo quali man mano che ci serviranno): questo menù ha il pregio di raccoglierli in un unico punto.

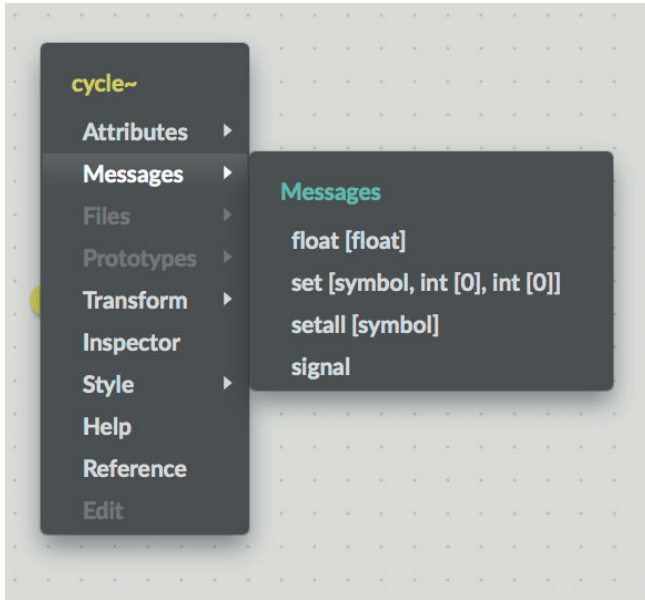


fig. 1.16c: *Object Action Menu*

PATCHER WINDOW TOOLBAR

Vediamo ora da vicino le quattro *Toolbar* che, come abbiamo detto, circondano la *Patcher Window*, ovvero la finestra dentro cui scriviamo i nostri programmi Max. La **Toolbar superiore** contiene gli oggetti disponibili in Max divisi per categorie (vedi 1.17) e un menù di Zoom nell'angolo in alto a sinistra tramite il quale è possibile ingrandire o rimpicciolire la visualizzazione della *patch*.

Le prime quattro icone dopo il menù Zoom corrispondono a quattro oggetti molto usati: l'*object box* (che già conosciamo), il *message box*, il *comment box* e l'oggetto `toggle` (faremo la conoscenza di questi ultimi oggetti tra poco).

Seguono cinque icone con cui è possibile aprire altrettante *palette* (tavolozze) contenenti oggetti di una determinata categoria:

Buttons: questa palette contiene gli oggetti interfaccia che fungono da pulsanti.

Numbers: contiene gli oggetti interfaccia che visualizzano e generano numeri (abbiamo già visto il *number box* in fig. 1.16).

Sliders: contiene oggetti interfaccia che funzionano come cursori, ovvero producono valori quando si trascina il mouse su di essi (abbiamo già visto questa *palette* in fig. 1.5).

Max for Live: contiene gli oggetti interfaccia Max for Live (ne parleremo nel secondo volume).

Add (altri oggetti): contiene tutti gli altri oggetti suddivisi in sotto-categorie (*Basic, Audio, Data, Images, Interface, Jitter*).

Abbiamo già visto la categoria *Audio* in fig. 1.7, ed esploreremo il contenuto delle altre categorie via via che ci serviranno.

C'è un'ultima icona nella *Toolbar* superiore che rappresenta un barattolo di vernice; questa icona non serve a richiamare degli oggetti, ma a impostare l'aspetto grafico della *Patcher Window* e degli oggetti Max: un clic sull'icona farà apparire la *Format Palette*. Non ci occuperemo dell'aspetto grafico di Max in questa prima parte; provate comunque a selezionare (in modalità *edit*) un *object box* nella patch 01_01.maxpat e, dopo aver fatto apparire la *Format Palette*, provate a modificarne il colore, oppure le dimensioni e il tipo di font (carattere) usato.

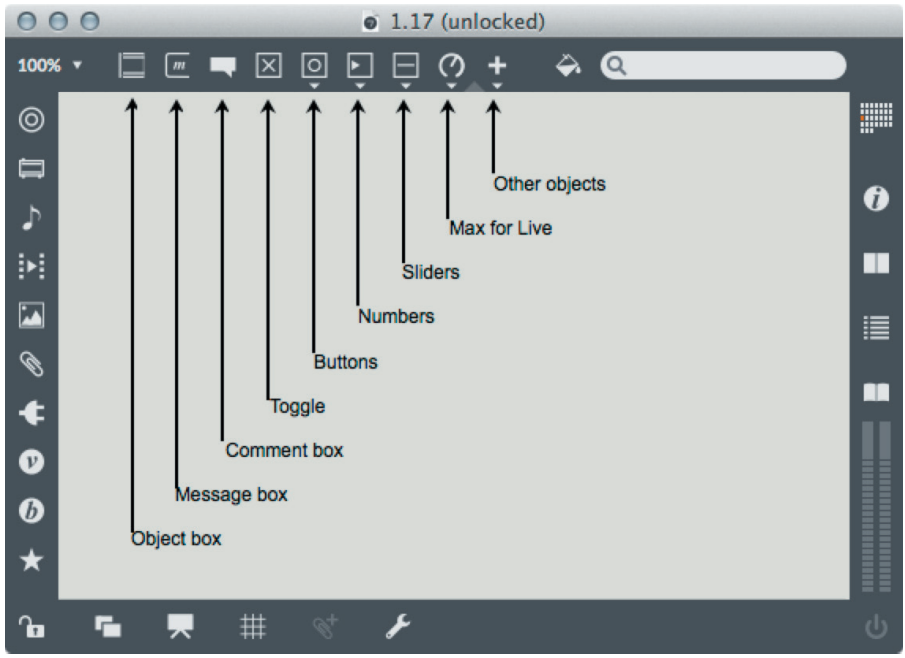


fig.1.17: la *Toolbar* superiore

La parte destra della *Toolbar* superiore è occupata da un campo di testo che ci permette di fare ricerche nella documentazione Max: provate ad esempio a cercare il termine "oscillator".

Tramite la **Toolbar di sinistra** è possibile accedere a tutti i file utili per la programmazione con Max (fig. 1.17b).

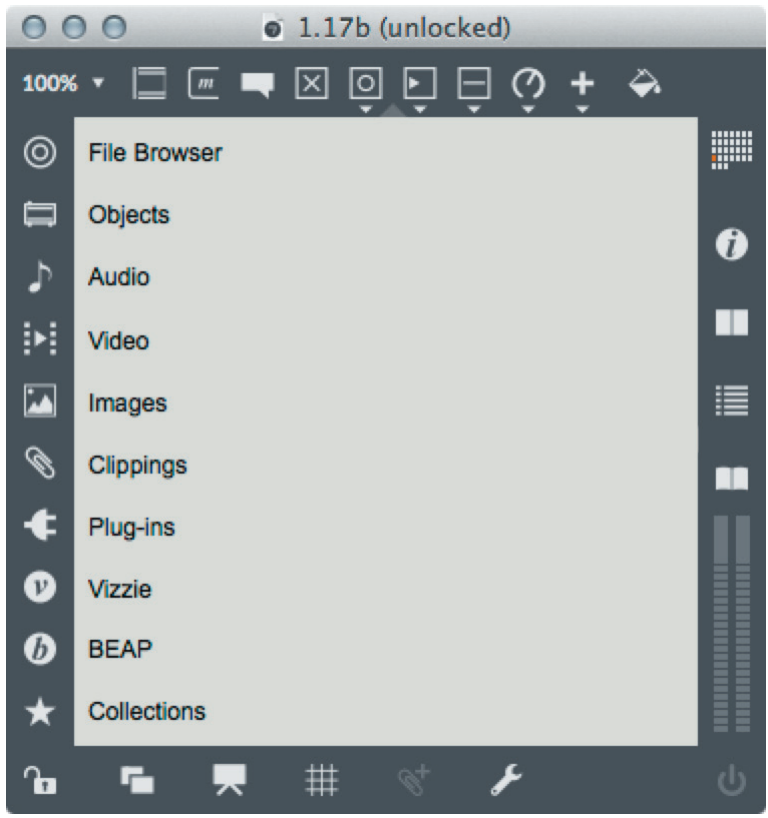


fig.1.17b: la *Toolbar* di sinistra

Saltiamo per il momento la prima icona, che apre il *File Browser* di cui parleremo verso la fine del capitolo, e facciamo clic sulla seconda: apparirà un menu laterale con tutti gli oggetti Max disponibili divisi per categorie (non solo gli oggetti interfaccia raggiungibili dalla *Toolbar* superiore). Un doppio clic sul nome dell'oggetto lo farà apparire all'interno della *patch*; in alternativa è possibile trascinare il nome dell'oggetto all'interno della *Patcher Window* (queste due modalità sono disponibili in tutti i menù della *Toolbar*).
La terza ci permette di richiamare file audio all'interno della *patch*, la quarta file video e la quinta immagini. Questi file devono essere inclusi nel *Percorso di Ricerca* di Max per essere visibili in questi menù (ci torneremo più avanti). Provate comunque, con il lucchetto aperto, a trascinare qualche file dal menù Audio all'interno di una *Patcher window* e osservate cosa succede; nel paragrafo 1.5 useremo questo menù.
La sesta icona (raffigurante una graffetta) ci permette di richiamare gli *Snippets*, frammenti di codice che possiamo memorizzare tramite la *Toolbar* inferiore (vedi sotto).

La settima icona ci dà l'accesso a tutti i plug-in Max for Live, VST e Audio Units che abbiamo installato nel nostro computer. Per il momento tralasciamo le icone successive.

La **Toolbar inferiore** contiene alcune funzioni utili per la programmazione di Max. La prima icona è quella del lucchetto che ci permette di passare, come sappiamo, dalla modalità *edit* alla modalità *performance* e viceversa.
Parleremo di altre icone contenute in questa *Toolbar* a tempo debito, per il momento occupiamoci della quinta icona, che rappresenta una graffetta molto simile a quella già vista nella *Toolbar* di sinistra e che ci permette di creare degli *Snippets*. Per capirne il funzionamento, aprite la *patch* 01_01.maxpat (fig. 1.9), andate in modalità *edit* e selezionate gli oggetti `gain~` (il *fader* verticale), `ezdac~` (il piccolo altoparlante), l'oggetto [`p` gain_to_amp][15]. A questo punto fate clic sull'icona della graffetta: apparirà una piccola finestra in cui vi si chiederà di assegnare un nome allo *snippet* da salvare, chiamatelo "audiostart" e battete invio. Ora create una nuova *patch* e fate clic sulla sesta icona della *Toolbar* di sinistra (l'altra graffetta): apparirà un menù che contiene, tra le altre cose, una voce "audiostart"; trascinate questo elemento nella *Patcher window*, comparirà lo *snippet* che avete appena salvato. Tramite questa tecnica è quindi possibile memorizzare dei pezzi di codice che sono riutilizzati spesso. Come vedete il menù contiene già degli *snippet* pronti per l'uso, e altri ne potrete creare durante il vostro lavoro con Max.

Passiamo alla **Toolbar di destra**: a parte la prima icona, che apre un calendario tramite il quale è possibile richiamare le *patch* usate recentemente, tutte le altre attivano la **Sidebar**, una specie di finestra "a cassetto" che si apre sul lato destro della *Patcher window*, e che può contenere una varietà di informazioni e funzioni utili.
Aprite ad esempio la *patch* 01_01.maxpat (fig. 1.9), andate in modalità *edit*, selezionate l'oggetto `ezdac~` (il piccolo altoparlante) e fate clic sulla seconda icona della *Toolbar* di destra (raffigurante le lettera "i" inscritta in un cerchio): si aprirà la *Sidebar* e ci mostrerà l'*inspector* dell'oggetto selezionato. L'*inspector* contiene tutti gli attributi che definiscono l'aspetto e il comportamento di un oggetto: ne parleremo dettagliatamente più avanti.
La terza icona ci permette di accedere alla documentazione dell'oggetto selezionato, mentre la quarta apre la *Max Console* che, come abbiamo visto, è una finestra tramite la quale Max ci mostra i messaggi di errore e altre informazioni utili che vedremo in seguito.
La quinta icona, infine, ci permette di accedere a una serie di lezioni interattive su Max.
Nella parte bassa della *Toolbar* sono visibili l'*Audio Meter/Gain*, con cui possiamo regolare il volume audio globale della *patch*, e il pulsante *Audio On/Off*, che ci permette di attivare disattivare la produzione di segnale audio della *patch* (esattamente come l'oggetto `ezdac~`).

[15] Per selezionare più oggetti potete usare la tecnica illustrata in fig. 1.10, oppure tenere premuto il tasto shift (maiuscole) e fare clic sugli oggetti uno dopo l'altro.

UN PO' DI ORDINE
Probabilmente avrete notato che alcune connessioni nella *patch* del file *01_01.maxpat* (figura 1.9) hanno degli angoli, sono divise in segmenti, ed hanno per questo un aspetto ordinato. Come si realizzano i cavi segmentati? Selezionando dal menù *Options* la voce **Segmented Patch Cords**.
Se usate questa opzione la procedura per connettere due oggetti sarà leggermente diversa: bisogna innanzitutto fare un clic sull'*outlet* che ci interessa, e "tirare" il cavo senza tenere premuto il tasto del mouse poiché il cavo stesso resterà "agganciato" al puntatore da solo. I segmenti si creano con un clic del mouse sul punto in cui vogliamo cambiare direzione: ad ogni clic si crea un nuovo segmento. L'ultimo clic lo faremo sull'*inlet* dell'oggetto da connettere. Se abbiamo fatto un errore e vogliamo liberarci di un cavo che è "agganciato" al puntatore del mouse dobbiamo pigiare <Mac: Command–clic> <Win: Control–clic> oppure il tasto *escape* (esc).
Se selezioniamo alcuni oggetti che sono allineati grosso modo orizzontalmente e digitiamo <Mac: Command–y> <Win: Control-Shift-a>, gli oggetti si allineeranno perfettamente. Lo stesso comando vale per incolonnare verticalmente degli oggetti posti l'uno sopra l'altro (i due `scope~` e i due `number~` del file *01_01.maxpat* sono stati incolonnati in questo modo). Gli oggetti inoltre si possono facilmente allineare grazie alla funzione **Snap to Object** che è attiva di *default*. In pratica ogni volta che spostiamo un oggetto nella *patch*, questo tenderà ad allinearsi all'oggetto più vicino.
Un'altra utile funzione è **Distribute**, reperibile nel menù *Arrange*: selezionando più oggetti è possibile distribuirli, appunto, a distanze uguali lungo una linea orizzontale o verticale (vedi fig. 1.18).

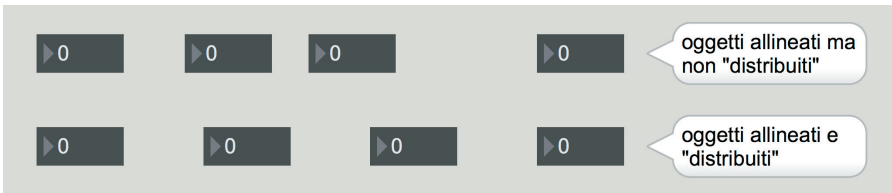


fig. 1.18: la funzione *Distribute*

Spesso inoltre una *patch* complessa può risultare molto affollata, con decine di oggetti e di cavi che si intrecciano: in questi casi si possono rendere invisibili alcuni cavi e oggetti in *performance mode* (mentre in *edit mode* restano, per ovvi motivi, sempre visibili). Per nascondere un oggetto o un cavo bisogna selezionarlo (in *edit mode*) e digitare <Mac: Command–k> <Win: Control–k>: passando alla modalità *performance* l'oggetto scomparirà. Per farlo riapparire, bisogna selezionarlo nuovamente in modalità *edit* e digitare <Mac: Command–l> <Win: Control–l>. Selezionando più oggetti è possibile nasconderli contemporaneamente con il comando già spiegato. In alternativa è possibile richiamare dal menù *Object* le voci **Hide on Lock**, per nascondere l'oggetto, e **Show on Lock**, per mostrarlo nuovamente. Provate a far sparire e riapparire gli oggetti della *patch* contenuta nel file *01_01.maxpat*. Un modo ancora più efficace di mettere ordine nelle *patch* è l'utilizzo della *presentation mode*: ne parleremo al paragrafo 1.3, dopo che avremo realizzato delle *patch* un po' più complesse.

ATTIVITÀ

Create una nuova *Patcher Window* e tentate di rifare la *patch* del file **01_01.maxpat**. Usate lo *snippet* che avete salvato nel precedente paragrafo (se non lo avete fatto, fatelo ora!), altrimenti vi sarà impossibile creare l'oggetto [`p` gain_to_amp]. Attenzione a non confondere l'oggetto `number~` con il *number box*! Se non riuscite a trovare gli oggetti grafici `scope~` e `number~` nella *Toolbar* superiore, ricordatevi che potete sempre prendere un *object box*, scrivere il nome dell'oggetto al suo interno e l'*object box* si trasformerà nell'oggetto grafico relativo. Noterete che la forma d'onda visualizzata dall'oscilloscopio creato da voi è diversa da quella del file originale. Vedremo perché nel prossimo paragrafo.

(...)

Il capitolo prosegue con:

1.2 AMPIEZZA, FREQUENZA E FORMA D'ONDA
Generatori limitati in banda

1.3 VARIAZIONI DI FREQUENZA E AMPIEZZA NEL TEMPO: INVILUPPI E GLISSANDI
Glissandi
Inviluppi
Presentation mode
Curve esponenziali e logaritmiche

1.4 RAPPORTO TRA FREQUENZA E INTERVALLO MUSICALE
Glissandi "naturali"
Conversione decibel-ampiezza

1.5 CENNI SULLA GESTIONE DEI FILE CAMPIONATI
Gestire i file con il file browser
Registrare un file audio
Leggere un suono da un buffer di memoria

1.6 CENNI SUL PANNING

1.7 ALTRE CARATTERISTICHE DI MAXMSP
Cavi neri o giallo/neri? Max vs MSP
Ordine di esecuzione degli oggetti Max
L'oggetto panel e il livello background

ATTIVITÀ

- SOSTITUZIONE DI PARTI DI ALGORITMI, CORREZIONE, COMPLETAMENTO E ANALISI DI ALGORITMI, COSTRUZIONE DI NUOVI ALGORITMI

VERIFICHE

- COMPITI UNITARI DI REVERSE ENGINEERING

SUSSIDI DIDATTICI

- LISTA COMANDI PRINCIPALI MAX - LISTA OGGETTI MAx - LISTA MESSAGGII, ATTRIBUTI E PARAMETRI PER OGGETTI MAX SPECIFICI - GLOSSARIO

Interludio A
PROGRAMMAZIONE CON MAX

CONTRATTO FORMATIVO

PREREQUISITI PER IL CAPITOLO

- Contenuti del Cap. 1 (teoria e pratica)

OBIETTIVI

Abilità

- Saper utilizzare tutte le funzioni di base del software Max riguardanti i numeri interi e con la virgola
- Saper generare e controllare sequenze di numeri casuali con possibile uso di metronomo
- Saper costruire algoritmi all'interno di subpatch e abstraction
- Saper gestire la replicazione di messaggi su varie uscite
- Saper gestire la composizione e scomposizione di liste anche mediante oggetti grafici
- Saper gestire l'utilizzo di argomenti variabili
- Saper gestire la comunicazione fra oggetti senza l'uso di cavi di collegamento virtuali

CONTENUTI

- I numeri integer e floating point in Max
- Generazione e controllo di numeri casuali con gli oggetti `random`, `drunk` etc.
- Generazione di eventi a ritmo regolare mediante l'oggetto `metro`
- Costruzione di subpatch e abstraction
- Gestione di liste e argomenti variabili
- Uso degli oggetti `send` e `receive` per comunicazione wireless fra gli oggetti

TEMPI - Cap. 1 (Teoria e Pratica) + Interludio A

Autodidatti

Per 300 ore globali di studio individuale (vol. I, teoria e pratica):

- ca. 100 ore

Corsi

Per un corso globale di 60 ore in classe + 120 di studio individuale (vol. I, teoria e pratica):

- ca. 16 ore frontali + 4 di feedback
- ca. 40 di studio individuale

ATTIVITÀ

Attività al computer:

- Sostituzione di parti di algoritmi, correzione, completamento e analisi di algoritmi, costruzione di nuovi algoritmi

VERIFICHE

- Compiti unitari di Reverse Engineering

SUSSIDI DIDATTICI

- Lista oggetti Max - Lista messaggi, attributi e parametri per oggetti Max specifici - Glossario

In questo primo "interludio" approfondiremo alcuni aspetti della programmazione con Max: si tratta di informazioni essenziali che ci saranno utili per la lettura del resto del libro. Vi consigliamo quindi di non saltarle, a meno che non siate già utenti esperti del programma. È altrettanto importante realizzare tutte le *patch* che vi verranno via via proposte. C'è da fare un piccolo sforzo che vi permetterà di ottenere grandi risultati.

IA.1 MAX E I NUMERI: GLI OPERATORI BINARI

Come ogni linguaggio di programmazione che si rispetti, anche Max può svolgere diverse operazioni sui numeri. Cominciamo col vedere gli operatori più semplici, quelli che ci permettono di fare addizioni, sottrazioni, moltiplicazioni e divisioni.

ADDIZIONI CON NUMERI INTERI

Ricreate la semplice *patch* di figura IA.1 (attenzione allo spazio tra "+" e "5"!).

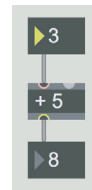


fig. IA.1: addizione

L'oggetto **+** somma il suo argomento (in questo caso 5) a qualunque numero riceva nel suo `inlet` sinistro: se mandiamo dei numeri all'oggetto tramite il *number box* superiore (facendo ad esempio clic sul *number box* in *performance mode* e utilizzando le freccette in basso a destra della tastiera alfanumerica per variare i numeri), possiamo vedere il risultato dell'operazione nel *number box* inferiore.
L'ingresso di destra di **+** serve a cambiare l'argomento: se inviamo un numero a questo ingresso tramite un nuovo *number box*, questo numero annulla e sostituisce l'argomento dell'oggetto **+** nelle somme che si produrranno mandando altri numeri all'ingresso di sinistra.
Aggiungete un *number box* a destra, come in fig. IA.2.

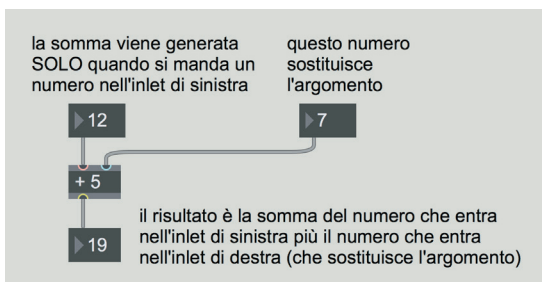


fig. IA.2: addizione con argomento variabile

Se effettuate un po' di prove, noterete che l'operazione viene eseguita solo quando entra un numero nell'ingresso di sinistra, non in quello di destra che serve invece a sostituire l'argomento.
Precisiamo meglio questo punto: i numeri che entrano nei due ingressi dell'oggetto + vengono memorizzati in due celle di memoria, che chiameremo "variabili interne", che si trovano nell'oggetto stesso. Ogni volta che un nuovo numero entra in uno dei due ingressi, la variabile interna corrispondente viene aggiornata; il vecchio numero viene cioè cancellato e sostituito con il nuovo. Oltre a ciò, quando il numero entra nell'ingresso sinistro viene eseguita l'operazione propria dell'oggetto +, cioè la somma del contenuto delle due variabili interne.
Questa è una caratteristica comune alla grande maggioranza degli oggetti Max che svolge la propria funzione solo quando un messaggio entra nell'ingresso di sinistra; i messaggi che entrano negli altri ingressi servono a modificare gli argomenti, ovvero ad aggiornare le variabili interne, o a modificare il comportamento degli oggetti.
Nel lessico degli oggetti Max, viene definito "caldo" l'ingresso di sinistra, che generalmente fa compiere l'operazione all'oggetto stesso (oltre ad aggiornare la variabile interna corrispondente), mentre vengono definiti "freddi" tutti gli altri ingressi che aggiornano le variabili interne senza produrre un output. Notate che il cerchio che appare all'ingresso di un oggetto che stiamo collegando in modalità *edit* è rosso per gli **ingressi "caldi"** e azzurro per gli **ingressi "freddi"**.
Ma c'è un modo per aggiornare l'addizione anche quando inviamo un nuovo numero all'ingresso destro del sommatore?
In altre parole, c'è un modo per far sì che un ingresso "freddo" si comporti come un ingresso "caldo"? Certo, lo possiamo vedere in fig. IA.3.

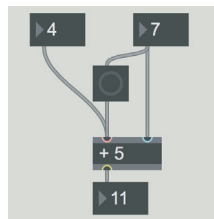


fig. IA.3: rendere "caldo" un ingresso "freddo"

Con questa *patch* possiamo mandare un numero all'ingresso destro e subito dopo un *bang* a quello sinistro (ricordate che l'ordine dei messaggi Max è da destra a sinistra): `button` infatti trasforma tutto ciò che riceve in un *bang*. Il messaggio *bang* come sappiamo forza l'oggetto che lo riceve a produrre un risultato: nel nostro caso l'addizione.
Che cosa viene addizionato? I numeri che l'oggetto ha nelle sue variabili interne: il numero che è stato appena inviato all'ingresso destro (nella figura è il numero 7) con l'ultimo numero che era stato inviato all'ingresso sinistro (in figura è il 4).[1]

[1] Vi ricordiamo che in questa figura, come anche nelle prossime, l'argomento "5" del sommatore viene sostituito dal numero che è stato inviato all'ingresso di destra.

In altre parole, con questo sistema anche l'ingresso "freddo" del sommatore si comporta come l'ingresso "caldo". Provate a ricostruire questa *patch* e verificate che il *number box* di destra ora produca un risultato ogni volta che viene modificato.
È essenziale che le posizioni degli oggetti in questa *patch* siano assolutamente le stesse: posizionare ad esempio `button` a destra dell'oggetto + può produrre risultati indesiderati, in quanto il *bang* verrebbe azionato prima che il nuovo numero entri nell'ingresso di destra dell'oggetto +, e quindi l'addizione si riferirebbe non al valore corrente del *number box* di destra, ma al suo ultimo valore prima che il *bang* venga azionato (vedi fig. IA.4).

fig. IA.4: risultato sbagliato per un errore di precedenza

Adesso cancellate i due *number box* superiori e collegate un *message box* all'ingresso sinistro dell'oggetto +, poi scrivete due numeri (separati da uno spazio) all'interno del *message box*, come in fig. IA.5.

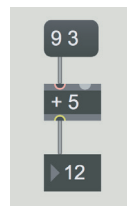


fig. IA.5: sommare una lista

Se ora fate clic (in *performance mode*) sul *message box* l'oggetto + sommerà i due numeri; l'operatore si comporta quindi come se avesse ricevuto il secondo numero nell'ingresso di destra (e come al solito l'argomento 5 che si trova all'interno dell'oggetto sommatore viene sostituito dal secondo operando).
Anche questa è una caratteristica comune a molti altri oggetti Max, e funziona anche con oggetti che hanno tre o più ingressi (che possono quindi accettare liste di tre o più elementi nell'ingresso sinistro e "smistarle" ai vari ingressi).

Vediamo una possibile applicazione musicale del sommatore Max: aprite il file **IA_01_trasposizione.maxpat** (fig IA.6).

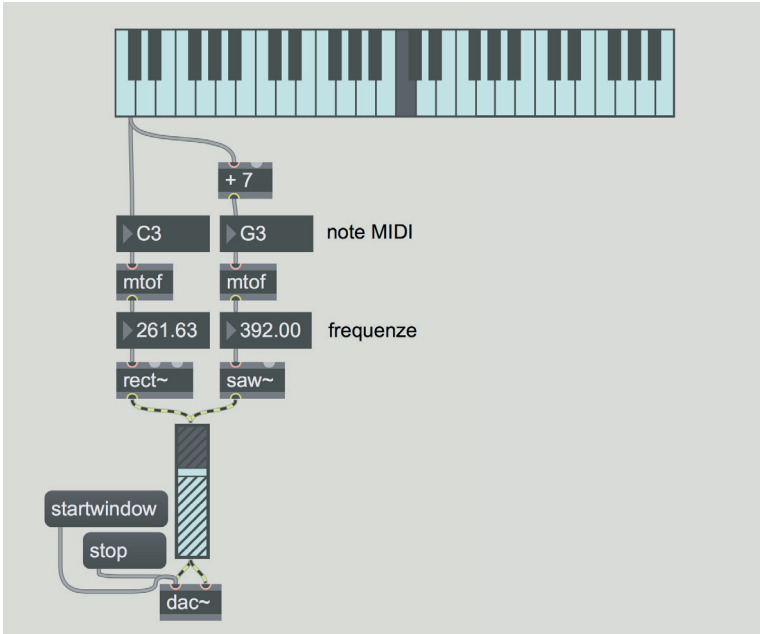


fig. IA.6: file IA_01_trasposizione.maxpat

In questa *patch*, ogni volta che viene premuto un tasto nell'oggetto `kslider`, vengono generate due note la seconda delle quali si trova 7 semitoni sopra la prima, ovvero una quinta sopra. In pratica, ogni volta che facciamo clic su un tasto di `kslider` il valore di nota MIDI corrispondente (ad esempio un DO, come in figura IA.6) viene inviato all'oggetto `mtof` di sinistra che lo converte in frequenza, ma viene anche inviato ad un sommatore che aggiunge il valore 7 alla nota (ottenendo ad esempio, con riferimento alla figura IA.6, un SOL): il risultato della somma viene infine inviato all'oggetto `mtof` di destra.
Tramite il sommatore, quindi, è possibile effettuare la trasposizione delle note MIDI a intervalli arbitrari (cioè a distanze fra due note che possiamo determinare noi). Dopo che i valori MIDI delle due note sono stati convertiti in valori frequenza, vengono inviati a due oscillatori, `rect~` e `saw~` che suonano l'intervallo di quinta. Provate a impostare nella *patch* di figura IA.6 altri intervalli (terza, ovvero 4 semitoni; quarta, ovvero 5 semitoni; ottava, ovvero 12 semitoni etc...), aggiungete poi un altro sommatore, collegato ad un altro `mtof` a sua volta collegato ad un oscillatore, e fate in modo che ad ogni tasto premuto nel `kslider` corrisponda un accordo maggiore di tre note: se, ad esempio, viene premuto il tasto C2 (il tasto che si trova un'ottava sotto il DO centrale), si ottiene il DO MI SOL della seconda ottava, cioè C2 E2 G2.

NUMERI CON LA VIRGOLA E ALTRE OPERAZIONI

Finora abbiamo usato solo numeri interi perché l'oggetto + quando ha un argomento intero (o nessun argomento) non gestisce i numeri con i decimali. Per poter fare delle operazioni con i decimali bisogna che il numero abbia una "virgola", che nella notazione anglosassone è in realtà un punto (cfr. fig. IA.7). In questo caso abbiamo collegato dei *float number box* all'operatore per poter usare i numeri con la virgola. Come vedete in figura l'argomento è "0." (ovvero uno 0 seguito da un punto, non è necessario mettere dei numeri dopo il punto), e la sua funzione è solo quella di comunicare all'oggetto + che intendiamo usare numeri non interi. Ricreate la *patch* e fate qualche somma con i numeri decimali (ricordatevi che l'ingresso "caldo" è solo quello di sinistra).

fig. IA.7: somma di numeri con la virgola

Tutto quello che abbiamo detto a proposito dell'addizione vale anche per la sottrazione, la moltiplicazione e la divisione; provate a ricostruire la *patch* di figura IA.8 e verificatene il funzionamento.

fig. IA.8: altri operatori matematici

Fare questi esercizi, per quanto facili possano sembrare, vi farà accorgere di tanti piccoli dettagli nel lavoro pratico con Max che possono rivelarsi importanti nelle operazioni che faremo sul suono nei prossimi capitoli.
Come vedete nel caso della "divisione con la virgola e uso del *message box*" abbiamo una lista di due numeri interi ("*10* e *4*"), ma dal momento che l'argomento che abbiamo dato all'operatore è un numero con la virgola il risultato contiene un numero decimale.
Vediamo come possiamo utilizzare alcuni di questi operatori in una applicazione musicale: come sappiamo un intervallo, cioè la distanza tra due note, può essere espresso in termini di rapporto di frequenza. Ad esempio un intervallo di quinta, corrispondente a 7 semitoni, può essere espresso con il rapporto 3/2: ciò vuol dire che, data una frequenza (ad esempio 261.63 Hertz, che corrisponde al DO centrale), se la moltiplichiamo per 3 e la dividiamo per 2, otteniamo la frequenza che si trova una quinta sopra (ovvero 392.44 Hz che corrisponde al SOL).

Aprite il file **IA_02_quinte.maxpat** (figura IA.9).

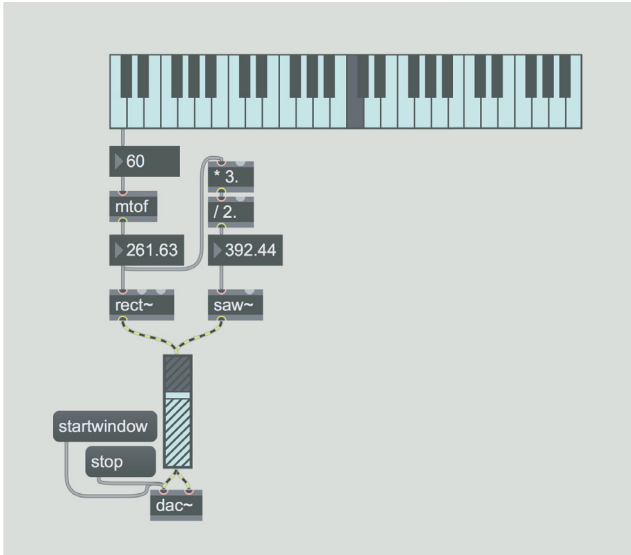


fig. IA.9: file IA_02_quinte.maxpat

Nella *patch* vediamo che la nota generata dall'oggetto `kslider` viene trasformata in frequenza tramite `mtof` e inviata a un oscillatore `saw~`; questa frequenza viene inoltre moltiplicata per 3/2, in modo da ottenere la frequenza che si trova una quinta sopra, e inviata a un secondo oscillatore `saw~`. Notate che gli argomenti della moltiplicazione e della divisione hanno un punto alla fine, in questo modo "comunichiamo" a Max di effettuare operazioni con i decimali. Fate attenzione ai due cavi che escono dal *float number box* che si trova sotto l'oggetto `mtof`: un cavo scende direttamente all'oggetto `saw~` che si trova immediatamente sotto, un secondo cavo piega a destra e risale fino al moltiplicatore con argomento 3.

Se confrontate le figure IA.6 e IA.9 noterete che la frequenza della nota che si trova una quinta sopra è leggermente diversa.
Tramite i due `mtof` infatti abbiamo calcolato un intervallo di quinta temperata (quella che si usa normalmente nella musica occidentale) che è esattamente di 7 semitoni temperati; tramite il rapporto 3/2 calcoliamo invece una quinta naturale che è più larga di quella temperata di circa 2 cents (centesimi di semitono temperato).

PDF

L'USO DEL PUNTO ESCLAMATIVO NELLE OPERAZIONI

Tutti gli operatori di cui abbiamo parlato sono operatori binari, così chiamati perché necessitano di due operandi, cioè di due numeri, per poter svolgere l'operazione. Negli oggetti che abbiamo visto il primo operando è il numero che entra nell'ingresso di sinistra, mentre il secondo operando è l'argomento (oppure è il numero che entra nell'ingresso di destra).

Per la sottrazione e la divisione esistono però altri due oggetti in cui è il secondo operando quello che entra nell'ingresso di sinistra (e che fa scattare l'operazione), mentre il primo operando è l'argomento. Il nome di questi oggetti è costituito dai simboli della sottrazione **!-** e della divisione **!/** preceduti da un punto esclamativo (vedi fig. IA.10).

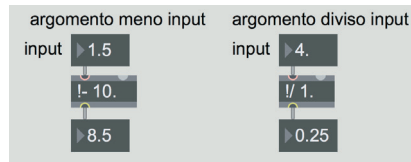


fig. IA.10: operatori con il punto esclamativo

Nel primo caso all'argomento 10 viene sottratto il valore in input 1.5, e il risultato è 8.5 (10-1.5 = 8.5); nel secondo caso l'argomento 1 viene diviso per il valore in input 4, e il risultato è 0.25, perché 1/4 = 0.25. In pratica i due operandi vengono invertiti rispetto alle normali sottrazioni e divisioni. Ricostruite la *patch* qui sopra e confrontate queste operazioni con le analoghe operazioni senza il punto esclamativo. Abbiamo già visto l'operando **!-** nella *patch* **01_17_pan.maxpat** nel paragrafo 1.6.
Di tutti questi operatori esiste una versione MSP che genera un segnale (+~, *~, /~, -~) e che abbiamo usato varie volte nel capitolo precedente. Gli operatori MSP devono obbligatoriamente ricevere almeno un segnale in uno dei due ingressi (normalmente quello di sinistra), mentre possono ricevere indifferentemente un segnale o un valore numerico nell'altro ingresso (oppure usare un argomento). Per avere altre informazioni sul funzionamento degli operatori vi ricordiamo che potete consultare l'help in linea facendo "*Alt-clic*" in modalità *edit* sull'oggetto che vi interessa.

GLI OGGETTI INT E FLOAT

Esistono due oggetti che ci permettono di memorizzare dei valori e di richiamarli successivamente con un *bang*: questi oggetti si chiamano **int** (per la memorizzazione di numeri interi) e **float** (per la memorizzazione di numeri con la virgola). In figura IA.11 vediamo che gli oggetti dispongono di due ingressi: se il valore viene inviato all'ingresso di sinistra (ingresso caldo), viene memorizzato e immediatamente trasmesso, se viene inviato all'ingresso di destra (ingresso freddo) viene solo memorizzato, e può essere trasmesso successivamente tramite un *bang* all'ingresso di sinistra. In entrambi i casi il valore resta nella memoria dell'oggetto (e può essere quindi richiamato con un *bang* tutte le volte che vogliamo) fino a che non si memorizza un nuovo valore.

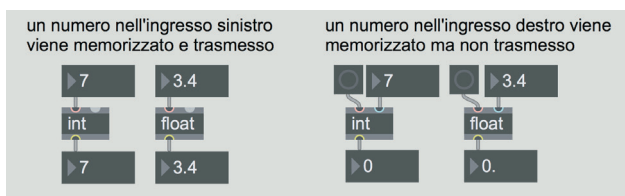


fig. IA.11: gli oggetti `int` e `float`

Il capitolo prosegue con:

IA.2 GENERAZIONE DI NUMERI CASUALI

IA.3 GESTIONE DEL TEMPO: METRO

IA.4 SUBPATCH E ABSTRACTION

Le subpatch
Le abstraction

IA.5 ALTRI GENERATORI RANDOM

IA.6 GESTIRE I MESSAGGI CON TRIGGER

IA.7 OGGETTI PER GESTIRE LE LISTE

L'oggetto unpack
L'oggetto pack
L'oggetto zl
Gli oggetti append e prepend
L'oggetto multislider

IA.8 IL MESSAGE BOX E GLI ARGOMENTI VARIABILI

Il messaggio di comando "set"
Gli argomenti variabili: il segno del dollaro ($)
Gli argomenti variabili: setdomain

IA.9 INVIARE SEQUENZE DI BANG: L'OGGETTO UZI

IA.10 SEND E RECEIVE

ATTIVITÀ

Attività al computer:

- Sostituzione di parti di algoritmi, correzione, completamento e analisi di algoritmi costruzione di nuovi algoritmi

VERIFICHE

- Compiti unitari di Reverse Engineering

SUSSIDI DIDATTICI

- Lista oggetti Max - Lista messaggi, attributi e parametri per oggetti Max specifici - Glossario

2T
SINTESI ADDITIVA E SINTESI VETTORIALE

CONTRATTO FORMATIVO

PREREQUISITI PER IL CAPITOLO

- Contenuti del cap. 1 (teoria)

OBIETTIVI

Conoscenze

- Conoscere la teoria della somma di onde (fase, interferenza costruttiva, distruttiva etc.)
- Conoscere la teoria e gli utilizzi di base della sintesi additiva a spettro fisso e spettro variabile, a spettro armonico e inarmonico
- Conoscere il rapporto fra fase e battimenti
- Conoscere il modo in cui si utilizzano le tabelle e in cui avviene l'interpolazione
- Conoscere la teoria e gli utilizzi di base della sintesi vettoriale

Abilità

- Saper individuare all'ascolto le caratteristiche di base dei suoni armonici e inarmonici
- Saper riconoscere all'ascolto i battimenti
- Saper individuare le varie fasi dell'inviluppo di un suono e saperne descrivere le caratteristiche

CONTENUTI

- Sintesi additiva a spettro fisso e variabile
- Suoni armonici e inarmonici
- Fase e battimenti
- Interpolazione
- Sintesi vettoriale

TEMPI - Cap. 2 (Teoria e Pratica)

Autodidatti

Per 300 ore globali di studio individuale (vol. I, teoria e pratica):

- ca. 60 ore

Corsi

Per un corso globale di 60 ore in classe + 120 di studio individuale (vol. I, teoria e pratica):

- ca. 10 ore frontali + 2 di feedback
- ca. 24 di studio individuale

ATTIVITÀ

Esempi interattivi

VERIFICHE

Test a risposte brevi
Test con ascolto e analisi

SUSSIDI DIDATTICI

Concetti di base - Glossario - Discografia - Un pò di storia

2.1 SINTESI ADDITIVA A SPETTRO FISSO

Il suono prodotto da uno strumento acustico o il suono in generale è propriamente un'oscillazione complessa esprimibile attraverso un insieme di più vibrazioni elementari che vengono prodotte contemporaneamente dallo strumento stesso: la somma di queste diverse vibrazioni contribuisce in modo determinante al timbro percepito e ne determina quindi la forma d'onda. Ogni suono può quindi essere scomposto in sinusoidi che, come abbiamo detto nel capitolo precedente, costituiscono il mattone fondamentale con cui è possibile costruire ogni altra forma d'onda. Ognuna di queste sinusoidi, o componenti del suono, ha una propria frequenza, ampiezza e fase. L'insieme delle frequenze, ampiezze e fasi delle sinusoidi che formano un determinato suono è definito **spettro sonoro**: vedremo più avanti come può essere rappresentato graficamente.
Non solo i suoni naturali, ma anche i suoni sintetici possono essere scomposti in un insieme di sinusoidi: le forme d'onda che abbiamo descritto nel paragrafo 1.2, quindi, hanno ciascuna un diverso spettro sonoro, ognuna contiene cioè una diversa mistura (combinazione) di sinusoidi (ad eccezione ovviamente della forma d'onda sinusoidale che contiene soltanto sé stessa!).

SPETTRO E FORMA D'ONDA

Spettro e forma d'onda sono due modi differenti per descrivere un suono. La forma d'onda è la rappresentazione grafica dell'ampiezza in funzione del tempo.[1] Nel grafico in figura 2.1 possiamo osservare la forma d'onda di un **suono complesso** in cui sull'asse delle *x* abbiamo il tempo, sull'asse delle *y* l'ampiezza; notiamo che la forma d'onda di questo suono è *bipolare*, cioè i suoi valori d'ampiezza oscillano al di sopra e al di sotto dello zero. Questa è una rappresentazione grafica nel **dominio del tempo** (*time domain*), vengono cioè tracciate tutte le ampiezze istantanee che vanno a formare, istante dopo istante la forma d'onda del suono complesso.

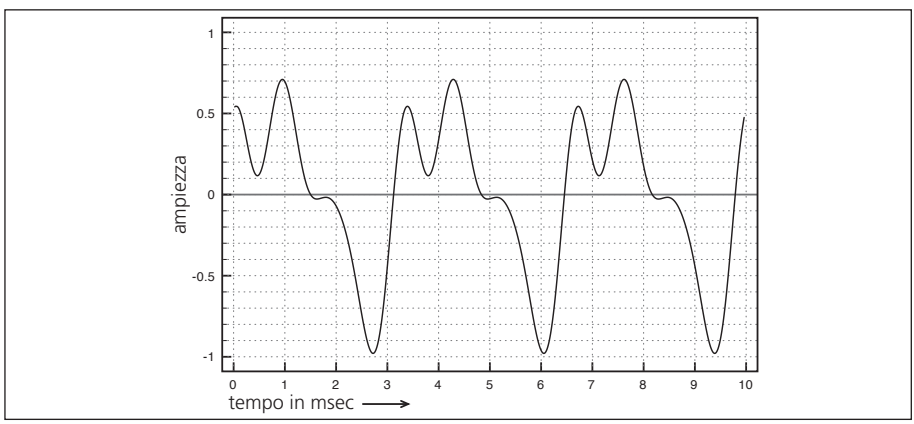


fig. 2.1: forma d'onda di un suono complesso

[1] Nel caso di suoni periodici la forma d'onda può essere rappresentata da un singolo ciclo.

Nella fig. 2.2a vediamo invece come si può scomporre in sinusoidi il suono complesso appena osservato: abbiamo infatti quattro sinusoidi con frequenza e ampiezza diverse, che sommate costituiscono il suono complesso rappresentato nel grafico precedente.

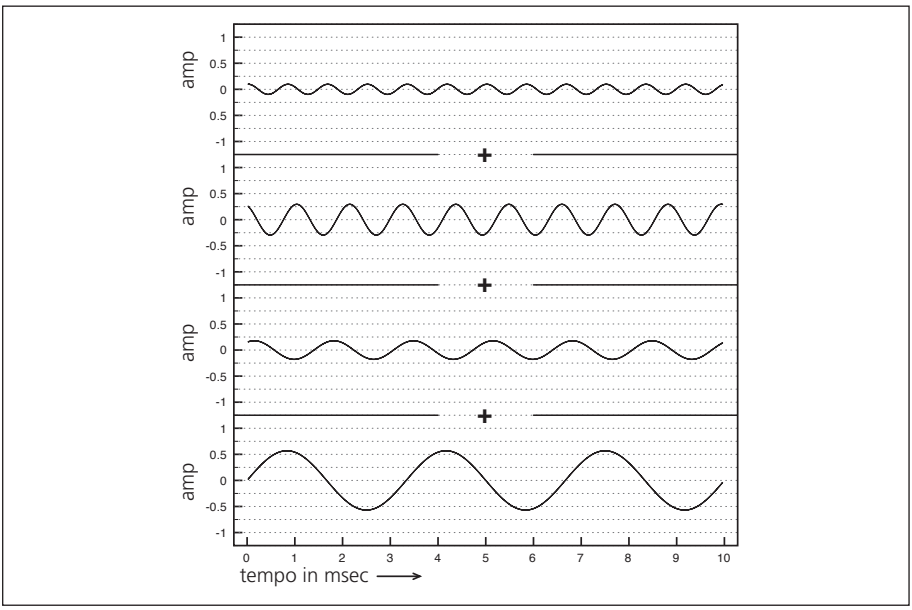


fig. 2.2a: scomposizione in sinusoidi di un suono complesso

Se vogliamo "fotografare" in modo più chiaro le varie frequenze di questo suono e la loro ampiezza, possiamo affidarci ad un grafico dello spettro sonoro, cioè la rappresentazione dell'ampiezza delle componenti in funzione della frequenza (ovvero nel **dominio della frequenza** o *frequency domain*). In questo caso sull'asse delle *x* abbiamo i valori di frequenza, sull'asse delle *y* i valori d'ampiezza. La figura 2.2b mostra le ampiezze di picco di ciascuna componente presente nel segnale.

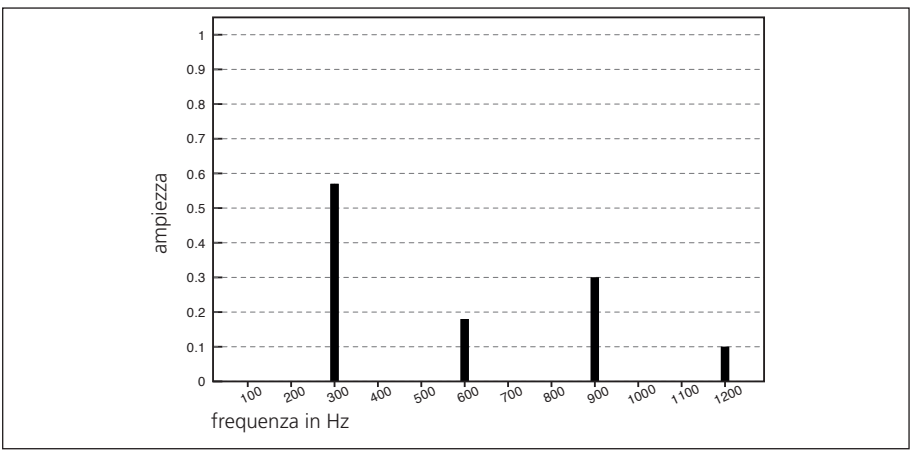


fig. 2.2b: spettro sonoro

Per vedere l'andamento nel tempo delle componenti di un suono possiamo usare anche il **sonogramma**, che ci mostra sull'asse delle *y* le frequenze e in quello delle *x* il tempo (fig. 2.2c). Le linee corrispondenti alle frequenze delle componenti sono più o meno marcate in funzione della loro ampiezza. In questo caso abbiamo 4 linee rette, perché si tratta di uno spettro fisso.

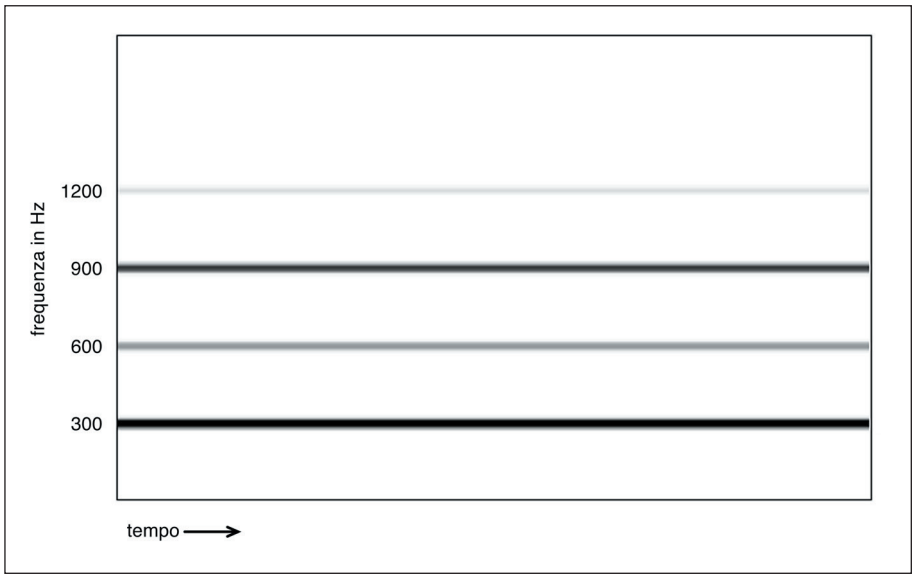


fig. 2.2c: sonogramma (o spettrogramma)

Adesso consideriamo il processo opposto, quello cioè in cui, invece di scomporre un suono complesso in sinusoidi, partiamo dalle singole sinusoidi e creiamo un suono complesso. La tecnica che ci permette di creare qualunque forma d'onda partendo dalla somma di sinusoidi si chiama **sintesi additiva** (fig. 2.3).

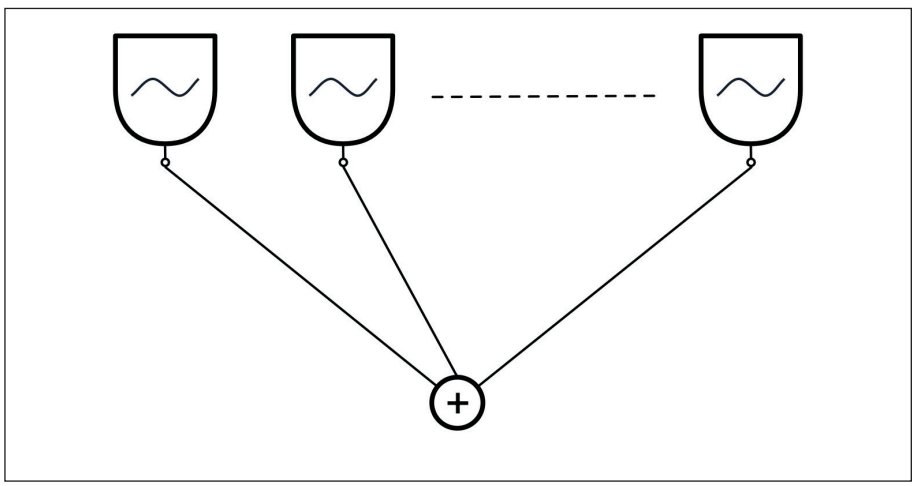


fig. 2.3: somma di segnali in uscita da oscillatori sinusoidali

In fig. 2.4 sono mostrate due onde sonore, A e B, e la loro somma C.

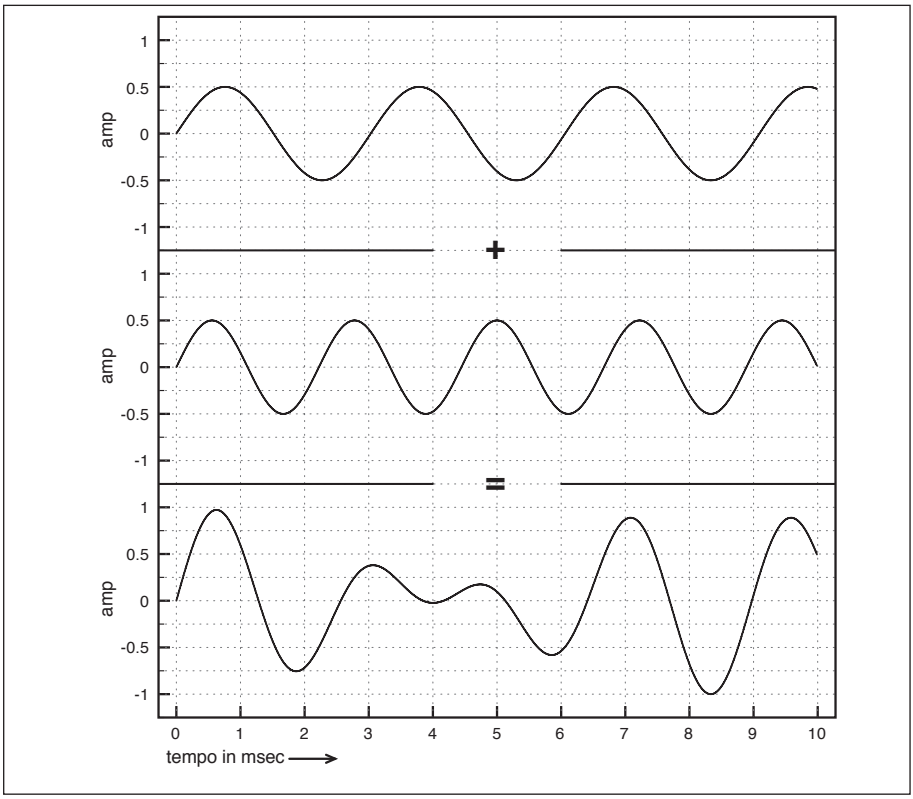


fig. 2.4: rappresentazione grafica di una somma di sinusoidi

Come si può facilmente verificare, le ampiezze istantanee dell'onda C sono ottenute sommando punto per punto le ampiezze istantanee dell'onda A a quelle dell'onda B. Istante per istante i valori dell'ampiezza delle diverse onde si sommano *algebricamente*, cioè con il loro segno, positivo o negativo. Nel caso in cui le ampiezze di A e B, in un dato istante, siano entrambe positive o entrambe negative, il valore assoluto di ampiezza di C risulterà maggiore di ciascuna delle singole componenti, si otterrà cioè un'**interferenza costruttiva**, ad es.

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

Nel caso in cui invece le ampiezze di A e B, in un dato istante, siano una positiva e l'altra negativa, il valore assoluto della loro somma algebrica C sarà minore di almeno una delle due componenti, e si avrà perciò un'**interferenza distruttiva**, ad es.

A = 0.3
B = -0.1
C = 0.2

"La maggior parte, anzi la quasi totalità, dei suoni che udiamo nel mondo reale non è composta da suoni *puri*, ma da **suoni complessi**, cioè suoni scomponibili in una maggiore o minore quantità di suoni puri; questi vengono detti *componenti* del suono complesso. Per meglio comprendere questo fenomeno, stabiliamo un'analogia con l'ottica. È noto come alcuni colori siano *puri*, cioè non ulteriormente scomponibili (rosso, arancio, giallo, fino al violetto). A ciascuno di essi corrisponde una certa *lunghezza d'onda* del raggio luminoso. Nel caso in cui sia presente soltanto uno dei colori puri, il prisma, che scompone la luce bianca nei sette colori dello spettro luminoso, mostrerà solamente quella componente. La medesima cosa avviene per il suono. A una certa **lunghezza d'onda**[2] del suono corrisponde una certa altezza percepita. Se non è presente contemporaneamente nessun'altra frequenza, il suono sarà *puro*. Un suono puro, come sappiamo, ha forma d'onda *sinusoidale*."
(Bianchini, R., Cipriani, A., 2001, pp. 69-70)

Le componenti di un suono complesso possono avere frequenze che sono multipli interi della frequenza della componente più grave. In questo caso la componente più grave si chiama **fondamentale** e le altre si chiamano **componenti armoniche** (ad esempio, fondamentale = 100 Hz, altre componenti = 200 Hz, 300 Hz, 400 Hz etc.) La componente che ha una frequenza doppia rispetto a quella della fondamentale si chiama *seconda armonica*; la componente di frequenza tripla della fondamentale viene detta *terza armonica* etc. Quando, come in questo caso, le componenti del suono sono multipli interi della fondamentale il suono si dice *armonico*. Notiamo che in un suono armonico la frequenza della fondamentale rappresenta il *massimo comun divisore* delle frequenze di tutte le altre componenti: è cioè il massimo numero che divide esattamente (senza resto) tutte le frequenze.

ESEMPIO INTERATTIVO 2A • *COMPONENTI ARMONICHE*

Se i suoni puri che compongono un suono complesso non sono multipli interi della componente più grave, abbiamo un suono inarmonico e le componenti prendono il nome di ...

(...)

[2] "La lunghezza di un ciclo si dice **lunghezza d'onda** e si misura in metri [m] o in centimetri [cm]. Questo è lo spazio che un ciclo occupa fisicamente nell'aria, e se il suono fosse visibile potrebbe facilmente essere misurato, per esempio con un metro." (Bianchini, R. 2003)

Il capitolo prosegue con:

La fase
Spettri armonici e inarmonici
Periodico/aperiodico, armonico/inarmonico
Interpolazione nella lettura di tabelle

2.2 BATTIMENTI

2.3 DISSOLVENZA INCROCIATA DI TABELLE:SINTESI VETTORIALE

2.4 SINTESI ADDITIVA A SPETTRO VARIABILE

ATTIVITÀ
ESEMPI INTERATTIVI

VERIFICHE
TEST A RISPOSTE BREVI
TEST CON ASCOLTO E ANALISI

SUSSIDI DIDATTICI
CONCETTI DI BASE - GLOSSARIO - DISCOGRAFIA - UN PÒ DI STORIA

2P
SINTESI ADDITIVA E SINTESI VETTORIALE

CONTRATTO FORMATIVO

PREREQUISITI PER IL CAPITOLO

- Cap. 1 (teoria e pratica), Cap.2 (teoria), Interludio A

OBIETTIVI

Abilità

- Saper sintetizzare un suono complesso partendo da semplici sinusoidi
- Saper sintetizzare suoni armonici e inarmonici in sintesi additiva e tabellare e trasformare gli uni negli altri e viceversa tramite controlli d'ampiezza e frequenza
- Saper realizzare forme d'onda triangolari, quadre, o a dente di sega approssimate mediante addizione di componenti armoniche sinusoidali
- Saper controllare i battimenti fra due suoni sinusoidali o armonici
- Saper sintetizzare suoni in sintesi vettoriale

Competenze

- Saper realizzare un breve studio sonoro basato sulle tecniche di sintesi additiva e memorizzarlo su file audio.

CONTENUTI

- Sintesi additiva a spettro fisso e variabile
- Suoni armonici e inarmonici
- Fase e battimenti
- Interpolazione
- Sintesi vettoriale

TEMPI - Cap. 2 (Teoria e Pratica)

Autodidatti

Per 300 ore globali di studio individuale (vol. I, teoria e pratica):

- ca. 60 ore

Corsi

Per un corso globale di 60 ore in classe + 120 di studio individuale (vol. I, teoria e pratica):

- ca. 10 ore frontali + 2 di feedback
- ca. 24 di studio individuale

ATTIVITÀ

- Attività al computer: sostituzione di parti di algoritmi, correzione, completamento e analisi di algoritmi, costruzione di nuovi algoritmi

VERIFICHE

- Realizzazione di uno studio breve
- Compiti unitari di Reverse Engineering

SUSSIDI DIDATTICI

- Lista oggetti Max - Lista messaggi, attributi e parametri per oggetti Max specifici - Glossario

2.1 SINTESI ADDITIVA A SPETTRO FISSO

Realizziamo per prima cosa una *patch* per la sintesi additiva con *spettro armonico*, ricreando con Max la figura 2.12 nel par 2.1 della parte teorica, che rappresenta 10 oscillatori le cui uscite vengono sommate da un miscelatore. Abbiamo bisogno quindi di 10 oggetti **`cycle~`** ai quali forniremo 10 frequenze, ciascuna delle quali deve essere un multiplo intero della fondamentale: per ottenere queste frequenze moltiplicheremo quindi una fondamentale data per una serie di interi successivi.
In figura 2.1 vediamo la *patch* relativa (**02_01_additiva.maxpat**).

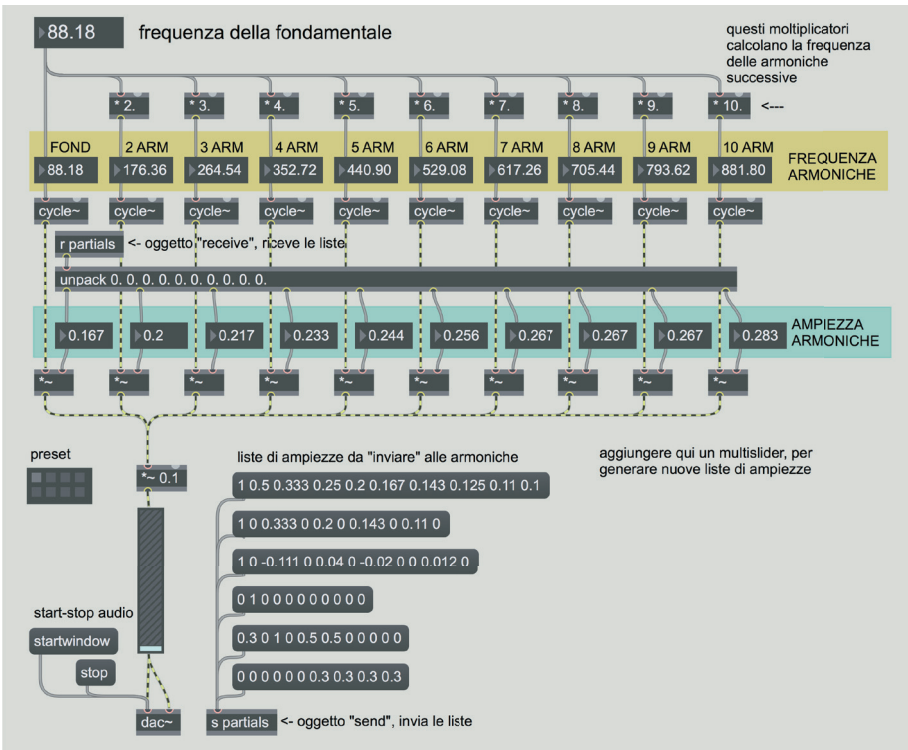


fig. 2.1: file 02_01_additiva.maxpat

Il *number box* in alto ci permette di impostare la frequenza fondamentale, che viene passata ad alcuni operatori **`*~`** che la moltiplicano per interi successivi in modo da darci una serie di frequenze multiple della fondamentale. La frequenza della fondamentale e le frequenze che escono dai moltiplicatori sono visualizzate in 10 *number box* collegati ad altrettanti **`cycle~`**, e ogni **`cycle~`** è connesso a un moltiplicatore di segnali (**`*~`**) che ne riscala l'uscita. Notiamo che normalmente l'ampiezza del segnale in uscita da **`cycle~`** ha valore 1, perciò il valore dato ad ogni moltiplicatore esprimerà direttamente l'ampiezza di ogni armonica (ad es. 1 • 0.5 = 0.5). Questi moltiplicatori di segnali, dunque, fanno sì che ogni armonica abbia una diversa ampiezza, a seconda del valore inserito nel moltiplicatore. Dato che abbiamo 10 oscillatori, ci sono dieci valori per i moltiplicatori da determinare.

Come abbiamo visto nelle *patch* al par. IA.10 dell'interludio A, possiamo inserire una lista di numeri in un *message box* oppure in un **`multislider`**.
Vediamo innanzitutto il primo caso: le ampiezze desiderate vengono raggruppate in liste all'interno di *message box* e vengono inviate ai 10 moltiplicatori da un oggetto **`send`**, [s partials], e dal corrispondente **`receive`**, che è collegato ad un **`unpack`** che distribuisce i singoli valori ai relativi moltiplicatori (fig. 2.1b).

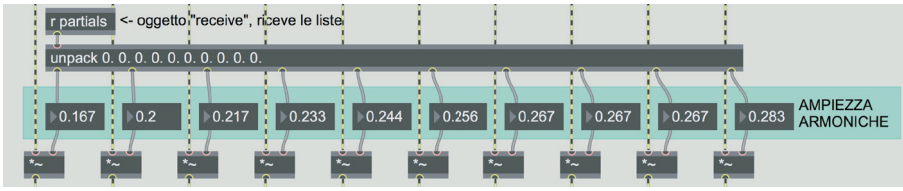


fig. 2.1b: l'oggetto **`unpack`** suddivide una lista nei singoli elementi

Le liste di ampiezze si possono creare anche utilizzando un **`multislider`**: createne uno in basso a destra, impostando tramite l'*inspector* 10 *slider* con *range* 0, 1.
Questo **`multislider`** dovrà essere collegato ad un oggetto **`send`** con argomento "partials".
Oltre a ciò è anche possibile agire direttamente sui *float number box* che sono collegati all'ingresso destro dei moltiplicatori di segnale per calibrare le diverse ampiezze, e memorizzare il risultato nell'oggetto **`preset`** a sinistra (facendo *shift-clic* in una casella). Dopo essere state riscalate dai moltiplicatori d'ampiezza, le componenti vengono inviate tutte a un ultimo moltiplicatore che riduce l'ampiezza complessiva ad un decimo.
Sappiamo infatti che due o più segnali che entrano in uno stesso ingresso si sommano; se tutte le componenti avessero ampiezza massima, cioè pari a 1, la forma d'onda risultante avrebbe un'ampiezza pari a 10.
L'ultimo moltiplicatore ha quindi lo scopo di riportare il valore assoluto dell'ampiezza tra 0 e 1 ed evitare così la distorsione del suono.[1]
Questo **`multislider`** deve inviare i suoi dati in modo continuo: durante la modifica di uno *slider*, cioè, tutte le variazioni devono essere inviate ai moltiplicatori. Di *default* però il **`multislider`** invia la sua lista soltanto alla fine della modifica, quando viene rilasciato il tasto del mouse.
Per ottenere l'aggiornamento continuo dobbiamo quindi richiamare l'*inspector* dell'oggetto, poi aggiungere un segno di spunta alla voce "Continuous Data Output when Mousing" (categoria "Style").

Create diversi profili spettrali, memorizzateli nei *preset* e cercate di apprezzare la differenza di timbro che si ha quando vengono enfatizzate le componenti più gravi, o le centrali, o quelle acute.

[1] Sulla distorsione dovuta ad ampiezze eccessive vedi par. 1.2 della teoria.

ATTIVITÀ

Usando la *patch* **02_01_additiva.maxpat** create due *preset*, uno con tutte le ampiezze delle armoniche dispari a 1 e le pari a 0 e l'altro con le armoniche pari a 1 e le dispari a 0. A parte il timbro, qual è la differenza più evidente? E perché?

Sempre con la *patch* **02_01_additiva.maxpat**, (facendo riferimento alla questione della fondamentale mancante o frequenza fantasma, trattata nel par. 2.1 della teoria) partite da uno spettro in cui tutte le ampiezze delle armoniche siano a 1, e poi azzerate l'ampiezza della fondamentale. Sentirete ancora uno spettro la cui fondamentale è pari alla frequenza che abbiamo azzerato. Provate a ridurre a zero l'ampiezza delle armoniche successive, una ad una. A che punto non si percepisce più la fondamentale?

LA FASE

Affrontiamo qui un argomento lungamente discusso nel par. 2.1 della parte di teoria, ovvero quello della *fase* di una forma d'onda periodica: si tratta di un concetto fondamentale che verrà sviluppato in diversi capitoli, è quindi importante comprendere come la fase viene gestita in Max.
Probabilmente avrete notato che l'oggetto `cycle~` ha due ingressi: il sinistro, come già sappiamo, ci permette di impostare la frequenza, il destro ci permette invece di impostare la fase *normalizzata*, ovvero ci permette di inviare un valore compreso tra 0 e 1 che corrisponde ad uno scostamento di fase compreso tra 0° e 360° (o tra 0 e 2π radianti, vedi box dedicato alla fase nel par. 2.1 della teoria). Ricostruite la *patch* di fig. 2.2 in una nuova *Patcher Window*.

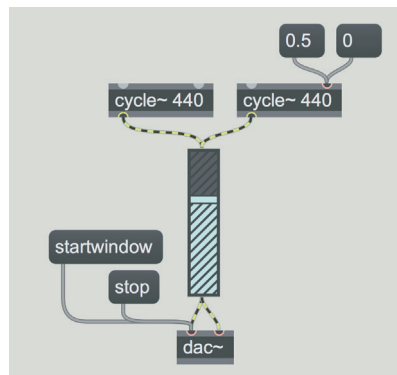


fig. 2.2: interferenza costruttiva e distruttiva

Abbiamo due `cycle~` a 440 Hz, il secondo dei quali ha due *message box* collegati all'ingresso destro (quello della fase, come sappiamo adesso) contenenti i numeri 0.5 e 0. Avviate la *patch* facendo clic su "*startwindow*" e alzate il cursore dell'oggetto `gain~`: si sente un La a 440 Hz generato dai due oscillatori che si sommano e sono perfettamente in fase.

Fate ora clic sul *message box* contenente il numero 0.5 e il suono scomparirà. Cosa è successo? È successo che abbiamo spostato la fase del secondo oscillatore di 0.5, ovvero di mezzo ciclo, o più propriamente di 180°. I due oscillatori si trovano quindi in *opposizione di fase*[2] e la somma dei due segnali è pari a 0. Facendo clic sul *message box* che contiene il numero 0, riportiamo i due oscillatori in concordanza di fase, e torniamo a sentire il suono.
Per comprendere meglio il meccanismo, modificate la *patch* come da fig. 2.3.

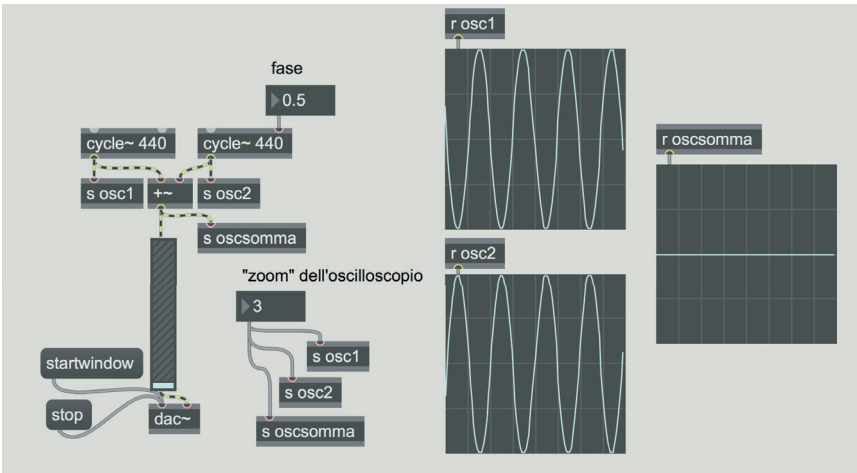


fig. 2.3: variazioni di fase

Abbiamo sostituito i due *message box* collegati all'ingresso destro del secondo oscillatore con un *float number box* e abbiamo aggiunto tre oggetti **`send`** per inviare i segnali prodotti dai due oscillatori e la loro somma a tre **`scope~`**. Inoltre, tramite altri tre **`send`** collegati ad un *number box* per i numeri interi, possiamo inviare l'attributo "samples per pixel"[3] ai tre oscilloscopi per regolare la visualizzazione. Notate che sia i segnali sia i valori numerici vengono inviati agli oscilloscopi tramite oggetti **`send`** con lo stesso nome ("osc1", "osc2" e "oscsomma"): l'oggetto **`scope~`** infatti, come sappiamo, accetta nello stesso ingresso segnali che visualizza come forma d'onda e valori numerici che utilizza come parametro per la visualizzazione.
Impostate i *number box* come quelli illustrati in figura e fate clic su "*startwindow*": dovreste vedere le onde sinusoidali in opposizione di fase nei due primi oscilloscopi e la somma "nulla" nel terzo. Fate clic su "*stop*" e verificate che le due sinusoidi negli oscilloscopi siano effettivamente in opposizione di fase (come si vede in figura). Proviamo ora (dopo aver nuovamente fatto clic su "*startwindow*") a modificare il valore del *float number box* che controlla la fase del secondo oscillatore: appena iniziamo a scendere da 0.5 a 0 udiamo un suono (corrispondente alla somma dei due oscillatori) che cresce progressivamente in

[2] O *polarità rovesciata*, vedi box dedicato alla fase nel par. 2.1 della teoria.
[3] Vedi par. 1.2, dove abbiamo spiegato che questo attributo può essere considerato, anche se un po' impropriamente, come una sorta di fattore di "zoom" dell'oscilloscopio.

ampiezza, fino ad oscillare tra -2 e 2 quando i due oscillatori sono in fase.[4] Se poi proviamo valori di fase superiori a 1, noteremo che quando la fase corrisponde a un numero intero il suono raggiunge la sua massima ampiezza, mentre si annulla quando la fase è a metà strada tra due numeri interi (1.5, 2.5, 3.5 etc.): valori superiori a 1 (ovvero superiori a 360°), infatti, fanno compiere ulteriori rotazioni alla fase come abbiamo spiegato nel paragrafo 2.1 della parte di teoria, nel box dedicato alla fase. Tutto questo funziona, naturalmente, solo se i due generatori hanno la stessa frequenza, in caso contrario il rapporto tra le fasi delle due oscillazioni varierebbe in continuazione, dando luogo, nel caso di frequenze solo leggermente diverse (ad esempio 440 e 441 Hz), al fenomeno dei battimenti trattato nel par. 2.2.
Vediamo ora un altro possibile utilizzo dell'ingresso di destra di `cycle~` (fig. 2.4).

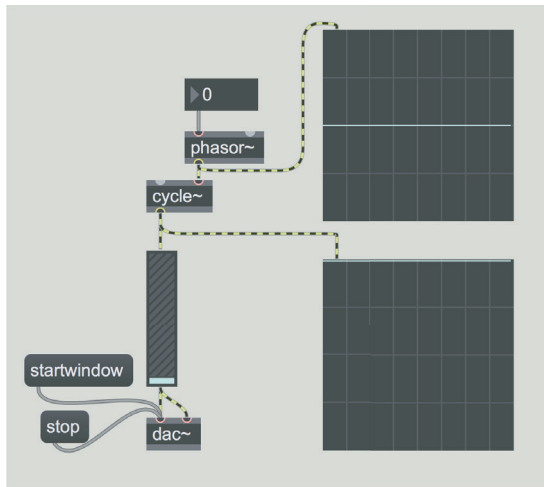


fig. 2.4: oscillatore guidato da un `phasor~`

Qui abbiamo un `cycle~` a 0 Hz (dal momento che non ha nessun argomento e nessun messaggio numerico all'ingresso di sinistra) la cui fase viene modificata da un `phasor~` che nell'immagine ha una frequenza di 0 Hz (è quindi fermo). Osservando gli oscilloscopi vediamo che `phasor~` genera un flusso di campioni di valore 0 (è quindi fermo all'inizio del suo ciclo), e `cycle~` genera un flusso di campioni di valore 1, è anch'esso quindi fermo all'inizio del suo ciclo. È importante sottolineare il fatto che il ciclo della forma d'onda generata da `cycle~` comincia da 1: come abbiamo infatti accennato nel par 1.1, `cycle~` genera una *cosinusoide*, non una sinusoide, e inizia quindi il suo ciclo partendo da 1 (cioè dal massimo valore positivo).[5]
Viceversa, come già sappiamo, l'oggetto `phasor~` inizia il suo ciclo partendo da 0.

[4] L'onda risultante oscilla tra -2 e 2 perché è la somma di due cosinusoidi (vedi più avanti) che oscillano tra -1 e 1 e sono perfettamente in fase.
[5] Per una definizione della funzione coseno vedi i parr. 1.2 e 2.1 della teoria.

Adesso diamo al **phasor~** una frequenza maggiore di 0, ad esempio 5 Hz (vedi fig. 2.5).

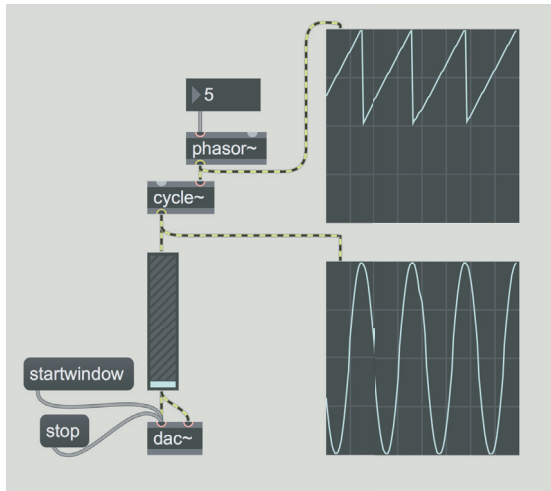


fig. 2.5: **phasor~** a 5 Hz

Provate a ricostruire questa *patch*: l'oggetto **phasor~** controlla in modo continuo la fase di **cycle~** facendolo oscillare alla sua stessa frequenza e con la sua stessa fase. Come sappiamo dal par. 1.2, infatti, **phasor~** genera rampe che vanno da 0 a 1: queste rampe, applicate alla fase di **cycle~**, ne provocano l'oscillazione (e questo ci spiega il motivo per cui l'oggetto **phasor~** si chiama così, perché una delle sue funzioni principali è guidare la fase di un altro oggetto).
Come vediamo in figura 2.5, ad ogni rampa generata da **phasor~** corrisponde un'oscillazione completa di **cycle~**: quest'ultimo quindi oscilla alla frequenza di 5 Hz. Se eliminiamo il **phasor~** e diamo a **cycle~** una frequenza di 5 Hz tramite un messaggio al suo ingresso di sinistra, o un argomento, otteniamo la stessa identica oscillazione: in altre parole è come se **cycle~** avesse un suo "motore phasor" interno.

(...)

Il capitolo prosegue con:

Uso di tabelle per gli oscillatori
Spettro fisso inarmonico

2.2 BATTIMENTI
Battimenti ritmici
Battimenti armonici

2.3 DISSOLVENZA INCROCIATA DI TABELLE:SINTESI VETTORIALE

2.4 SINTESI ADDITIVA A SPETTRO VARIABILE
Gestione degli inviluppi delle componenti tramite
Interfaccia grafica
Gestione degli inviluppi delle singole componenti
Tramite testo
Uso di banchi di oscillatori
Conversione da millisecondi a campioni e viceversa
Spettri variabili con banchi di oscillatori
Il controllo mediante mascheratura

ATTIVITÀ
- Attività al computer: sostituzione di parti di algoritmi, correzione, completamento e analisi di algoritmi, costruzione di nuovi algoritmi

VERIFICHE
- Realizzazione di uno studio breve
- Compiti unitari di Reverse Engineering

SUSSIDI DIDATTICI
- Lista oggetti Max - Lista attributi e parametri per oggetti Max specifici - Glossario

3T
GENERATORI DI RUMORE, FILTRI E SINTESI SOTTRATTIVA

CONTRATTO FORMATIVO

PREREQUISITI PER IL CAPITOLO

- Contenuti dei capp. 1 e 2 (teoria)

OBIETTIVI

Conoscenze

- Conoscere la teoria della sintesi sottrattiva
- Conoscere la teoria e l'uso dei parametri dei filtri principali
- Conoscere la differenza fra la teoria dei filtri ideali e la risposta di quelli digitali
- Conoscere la teoria e la risposta dei filtri FIR e IIR
- Conoscere l'uso di filtri disposti in serie o in parallelo
- Conoscere la teoria e l'uso degli equalizzatori grafici e parametrici
- Conoscere le applicazioni dei filtri a diversi tipi di segnale
- Conoscere le funzioni principali di un tipico sintetizzatore per sintesi sottrattiva

Abilità

- Saper individuare all'ascolto le caratteristiche di base di un filtraggio e saperle descrivere

CONTENUTI

- Sintesi sottrattiva
- Filtri passa-basso, passa-alto, passa-banda ed elimina-banda
- Filtri high shelving, low shelving, peak/notch
- Fattore Q
- Ordini dei filtri
- Filtri finite impulse response e infinite impulse response
- Equalizzatori grafici e parametrici
- Filtraggio di suoni provenienti da generatori di rumore, suoni campionati, impulsi

TEMPI - Cap. 3 (Teoria e Pratica) + Interludio B

Autodidatti

Per 300 ore globali di studio individuale (vol. I, teoria e pratica):

- ca. 110 ore

Corsi

Per un corso globale di 60 ore in classe + 120 di studio individuale (vol. I, teoria e pratica):

- ca. 18 ore frontali + 4 di feedback
- ca. 44 di studio individuale

ATTIVITÀ

- Esempi interattivi

VERIFICHE

- Test a risposte brevi
- Test con ascolto e analisi

SUSSIDI DIDATTICI

- Concetti di base - Glossario - Discografia - Un po' di storia

In questo capitolo parleremo dei *filtri*, argomento fondamentale nel campo del sound design e della musica elettronica, e di una tecnica di sintesi che ne fa largo uso: la sintesi sottrattiva.

3.1 SORGENTI PER LA SINTESI SOTTRATTIVA

La **sintesi sottrattiva** nasce dall'idea di poter creare un suono modificando, attraverso l'uso di filtri, l'ampiezza di alcune componenti di un altro suono. Lo scopo della maggior parte dei filtri digitali, infatti, è quello di alterare in qualche modo lo spettro di un suono. Un **filtro** quindi è un dispositivo che agisce prevalentemente su alcune frequenze contenute in un segnale, di solito attenuandole o enfatizzandole.[1]
Qualsiasi suono può essere filtrato. Attenzione, però! Non possiamo attenuare o enfatizzare componenti che non esistono nel suono originario, ad esempio non ha senso usare un filtro per enfatizzare i 50 Hz se stiamo filtrando la voce di un soprano, semplicemente perché tale frequenza non è presente nel suono originario.
I suoni originari utilizzati in genere nella sintesi sottrattiva sono ricchi dal punto di vista spettrale, e, come abbiamo detto, i filtri servono a modellare lo spettro di questi suoni e ottenere in uscita suoni diversi da quelli originari.
In questo paragrafo ci concentreremo sui suoni tipici utilizzati come sorgenti per la sintesi sottrattiva e per l'uso dei filtri in genere. Affronteremo le tecniche di filtraggio nei paragrafi successivi.

In generale nel lavoro in studio i filtri vengono utilizzati con diversi tipi di suoni:

- Suoni provenienti da generatori di rumore, da generatori di impulsi, da banchi di oscillatori, da altri generatori di segnale e da algoritmi di sintesi
- File audio/suoni campionati
- Suoni provenienti da fonti dal vivo in tempo reale (per esempio un suono proveniente dal microfono di un musicista che sta suonando un oboe)

GENERATORI DI RUMORE: RUMORE BIANCO E RUMORE ROSA

Uno dei suoni più utilizzati come sorgente per la sintesi sottrattiva è il **rumore bianco**, cioè un suono che contiene tutte le frequenze udibili e il cui spettro è essenzialmente piatto (pur essendo l'ampiezza delle singole frequenze distribuita casualmente).
È chiamato rumore bianco in analogia con l'ottica, dato che il colore bianco contiene tutti i colori dello spettro visibile. Si utilizza spesso il rumore bianco come suono originario perché, dal momento che contiene tutte le frequenze udibili, può essere utilmente filtrato da qualunque tipo di filtro a qualunque frequenza (vedi fig. 3.1).

[1] Oltre all'ampiezza un filtro può modificare la fase delle componenti di un suono.

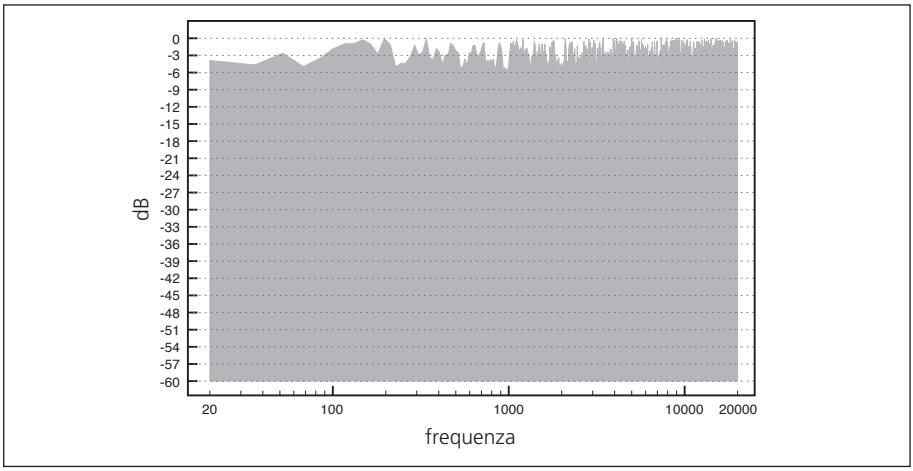


fig. 3.1: spettro del rumore bianco

Un altro tipo di rumore spesso utilizzato nella sintesi sottrattiva è il **rumore rosa**. Quest'ultimo, a differenza del rumore bianco, ha uno spettro la cui energia diminuisce all'aumentare della frequenza, più precisamente l'attenuazione è di 3 dB per ottava[2]; è anche chiamato generatore di rumore 1/f, per indicare che la sua energia spettrale è proporzionale al reciproco della frequenza. Viene spesso utilizzato, insieme ad un analizzatore di spettro, per testare e correggere la risposta in frequenza di impianti audio in relazione ad ambienti in cui si svolgono eventi musicali (fig. 3.2).

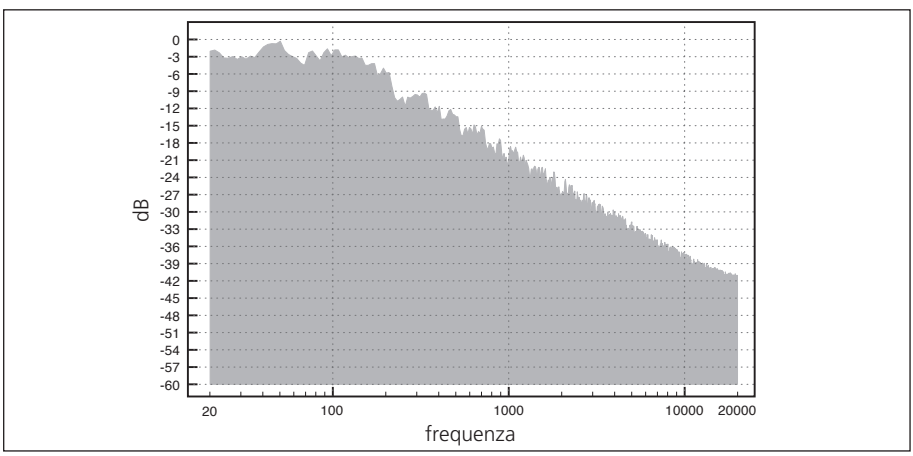


fig. 3.2: spettro del rumore rosa

[2] Un altro modo per definire la differenza tra rumore bianco e rumore rosa è questo: mentre lo spettro del rumore bianco ha la stessa energia per ogni frequenza, lo spettro del rumore rosa ha *la stessa energia per ogni ottava*. Dal momento che salendo verso l'acuto un'ottava occupa uno spazio di frequenza doppio dell'ottava precedente, la stessa energia totale viene distribuita in uno spazio sempre maggiore e questo determina l'attenuazione di 3 dB caratteristica del rumore rosa.

Nei sistemi digitali in genere il rumore bianco viene prodotto mediante generatori di numeri *random* (casuali): l'onda casuale che ne deriva contiene tutte le frequenze riproducibili dal sistema digitale usato. In realtà, i generatori di numeri *random* utilizzano procedure matematiche che non sono propriamente casuali: infatti generano serie che si ripetono dopo un certo numero di eventi. Questi generatori vengono perciò definiti **generatori pseudocasuali**.
Modificando alcuni parametri si possono generare tipi di rumore diversi. Il generatore di rumore bianco, ad esempio, genera campioni casuali alla frequenza di campionamento (ovvero, se la frequenza di campionamento del sistema è 48000 Hz, vengono generati 48000 valori casuali al secondo); è possibile però modificare la frequenza con cui questi numeri vengono generati: utilizzando una frequenza di generazione dei numeri pari a 5000 al secondo ad esempio, non avremo più un rumore bianco, ma un rumore con forte attenuazione sulle frequenze acute.

Quando la frequenza di generazione dei campioni è minore della frequenza di campionamento si crea il problema di come "riempire i vuoti" tra un campione e il successivo: un sistema DSP (vedi glossario cap. 1T) infatti deve sempre produrre un numero di campioni al secondo pari alla frequenza di campionamento.
Ci sono vari modi per risolvere questo problema. In particolare possiamo fare tre esempi:

- **Generatori di campioni pseudocasuali semplici**.
  Generano valori casuali ad una frequenza data e mantengono il valore di ciascun campione fino alla generazione del valore successivo, creando un andamento a gradini. In fig. 3.3 vediamo il grafico relativo ad un generatore di rumore a 100 Hz: ogni valore casuale generato viene ripetuto nei campioni successivi per un periodo pari ad 1/100 di secondo, dopo di che si genera un nuovo valore. Se la frequenza di campionamento fosse 48000 Hz, ad esempio, ogni valore casuale verrebbe ripetuto per 48000/100 = 480 campioni.

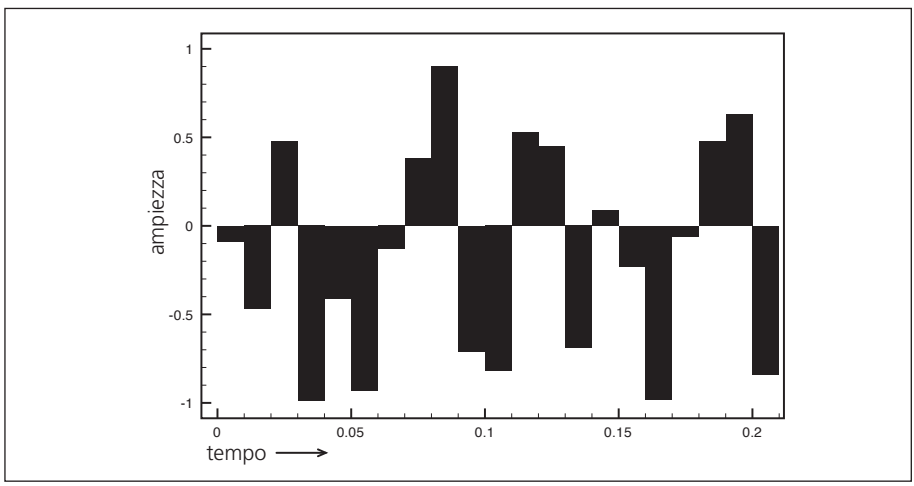


fig. 3.3: generazione di valori pseudo-casuali

- **Generatori di campioni pseudocasuali con interpolazione** fra un numero *random* e il successivo (vedi la sezione sull'interpolazione lineare nel cap. 2.1): come possiamo vedere in fig. 3.4 i campioni che si trovano fra i valori casuali prodotti dal generatore formano un segmento di retta che porta gradualmente da un valore all'altro.

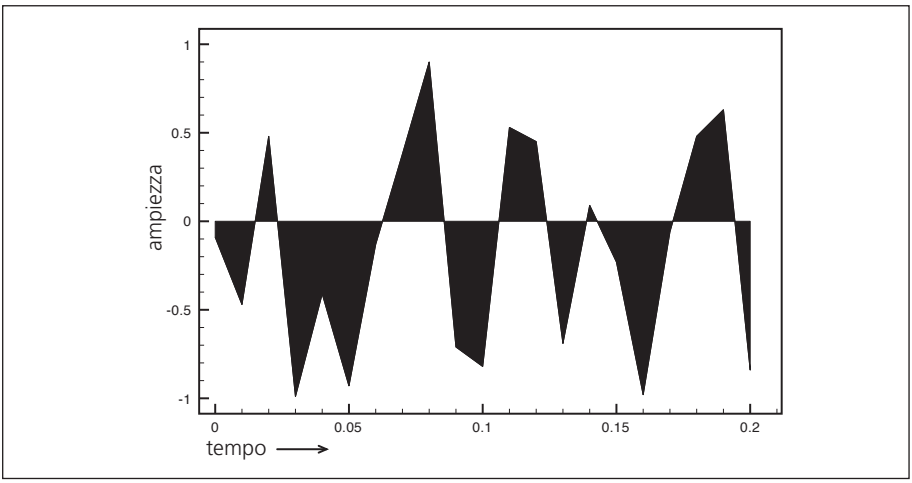


fig. 3.4 : generazione di valori pseudo-casuali con interpolazione lineare

L'interpolazione tra un valore e l'altro può essere lineare, come quella illustrata in figura, oppure *polinomiale*, realizzata cioè utilizzando delle funzioni polinomiali (che non approfondiremo in questa sede) che collegano i valori tramite delle curve anziché tramite delle rette (vedi fig. 3.5). Le interpolazioni polinomiali più usate nella computer music sono l'interpolazione *quadratica* (realizzata cioè con un polinomio di secondo grado) e quella *cubica* (polinomio di terzo grado): normalmente i linguaggi di programmazione per la sintesi e l'elaborazione del suono hanno già degli algoritmi pronti che realizzano queste interpolazioni.

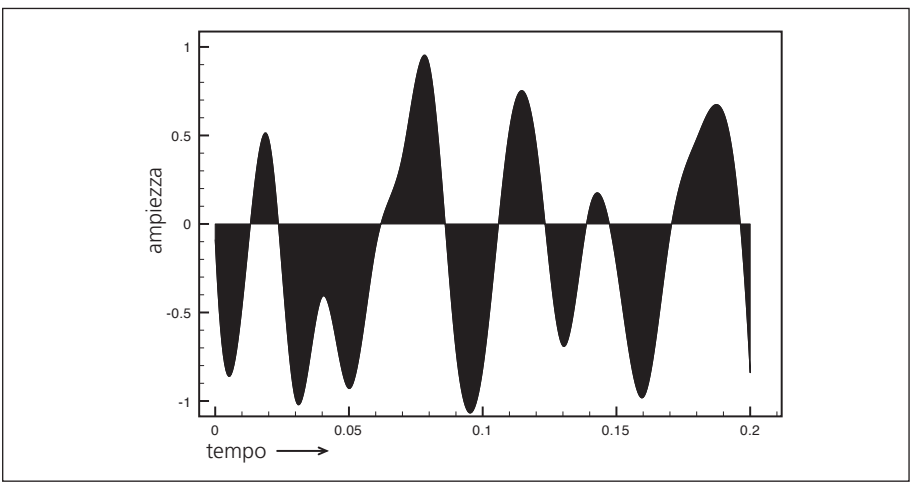


fig. 3.5: generazione di valori pseudo-casuali con interpolazione polinomiale

- **Generatori di campioni pseudocasuali con filtro:** in questo tipo di sistema il segnale in uscita viene filtrato mediante un filtro passa-basso. Parleremo di questo tipo di generatore nella sezione dedicata ai filtri passa-basso.

ESEMPIO INTERATTIVO 3A • *GENERATORI DI RUMORE* - PRESET 1-4

OSCILLATORI E ALTRI GENERATORI DI SEGNALE

Nel paragrafo 1.2 abbiamo visto alcune forme d'onda "classiche" impiegate normalmente nei sintetizzatori, come l'onda quadra, l'onda a dente di sega e la triangolare; nel paragrafo 2.1, inoltre, abbiamo visto che tali forme d'onda, quando sono geometricamente perfette (cioè perfettamente quadrate, triangolari etc.) contengono un numero infinito di componenti. La presenza di infinite componenti, però, causa alcuni importanti problemi nella riproduzione del suono digitale: ciò è dovuto al fatto che una scheda audio non può riprodurre frequenze superiori alla metà della frequenza di campionamento[3] (approfondiremo questo argomento nel capitolo 5).
Quando si tenta di riprodurre digitalmente un suono in cui la frequenza delle componenti supera le capacità della scheda, si ottengono componenti indesiderate quasi sempre inarmoniche. Per evitare questo problema vengono spesso usati, nella musica digitale, gli **oscillatori limitati in banda**. Tali oscillatori, che riproducono le forme d'onda classiche, sono fatti in modo che le componenti non superino mai la metà della frequenza di campionamento. I suoni generati da questo tipo di oscillatori possono quindi essere un buon punto di partenza per ottenere sonorità particolari mediante l'uso di filtri, e sono largamente impiegati, infatti, nell'implementazione di sintetizzatori che operano mediante la sintesi sottrattiva. Nel par. 3.5 analizzeremo la struttura di un tipico sintetizzatore sottrattivo.
Naturalmente sarà possibile utilizzare per la sintesi sottrattiva anche suoni sintetici ricchi di parziali realizzati con le diverse tecniche (ad esempio le tecniche di sintesi non lineare o la sintesi per modelli fisici) di cui parleremo nei prossimi capitoli.

FILTRAGGIO DI SUONI CAMPIONATI

Uno degli utilizzi più comuni dei filtri e degli equalizzatori, al là della sintesi sottrattiva, è il filtraggio dei suoni campionati. A differenza del rumore bianco, che contiene tutte le frequenze alla stessa ampiezza, un suono campionato conterrà un numero limitato di frequenze e rapporti d'ampiezza fra le componenti che possono variare a seconda del suono considerato.

[3] È per questo motivo che la frequenza di una scheda audio è quasi sempre superiore al doppio della massima frequenza udibile dall'uomo.

È necessario perciò, prima di effettuare un filtraggio, essere coscienti del *range* frequenziale del suono da elaborare. Infatti, come già accennato in precedenza, *possiamo attenuare o esaltare solo frequenze che siano già contenute nel file di partenza*. Ciò vale anche per i suoni che giungono al nostro computer da una fonte dal vivo.

(...)

Il capitolo prosegue con:

ATTIVITÀ

- ESEMPI INTERATTIVI

VERIFICHE

- TEST A RISPOSTE BREVI
- TEST CON ASCOLTO E ANALISI

SUSSIDI DIDATTICI

- CONCETTI DI BASE - GLOSSARIO - DISCOGRAFIA - UN PO' DI STORIA

3P
GENERATORI DI RUMORE, FILTRI E SINTESI SOTTRATTIVA

CONTRATTO FORMATIVO

PREREQUISITI PER IL CAPITOLO

- Contenuti dei capp. 1 e 2 (teoria e pratica), cap.3 (teoria), Interludio A

OBIETTIVI

Abilità

- Saper generare e controllare diversi tipi di segnali complessi per la sintesi sottrattiva (rumore bianco, rumore rosa, impulsi etc.)
- Saper costruire algoritmi con filtri passa-basso, passa-alto, passa-banda, elimina-banda, filtri shelving e filtri risonanti, e controllarne, fra i vari parametri, anche il Q e l'ordine dei filtri.
- Saper costruire filtri FIR o non ricorsivi e filtri IIR o ricorsivi
- Saper costruire semplici sintetizzatori in sintesi sottrattiva
- Saper scrivere algoritmi con collegamenti in serie e in parallelo dei filtri
- Saper costruire equalizzatori grafici e parametrici

Competenze

- Saper realizzare un breve studio sonoro basato sulle tecniche di sintesi sottrattiva e memorizzarlo su file audio.

CONTENUTI

- Sorgenti per la sintesi sottrattiva
- Filtri passa-alto, passa-basso, passa-banda, elimina-banda, filtri shelving e filtri risonanti
- Il Quality Factor e l'ordine dei filtri
- Filtri FIR e IIR
- Il collegamento in serie e in parallelo dei filtri
- Gli equalizzatori grafici e parametrici

TEMPI - Cap. 3 (Teoria e Pratica) + Interludio B

Autodidatti

Per 300 ore globali di studio individuale (vol. I, teoria e pratica):

- ca. 110 ore

Corsi

Per un corso globale di 60 ore in classe + 120 di studio individuale (vol. I, teoria e pratica):

- ca. 18 ore frontali + 4 di feedback
- ca. 44 di studio individuale

ATTIVITÀ

- Attività al computer: sostituzione di parti di algoritmi, correzione, completamento e analisi di algoritmi, costruzione di nuovi algoritmi

VERIFICHE

- Realizzazione di uno studio breve
- Compiti unitari di Reverse Engineering

SUSSIDI DIDATTICI

- Lista oggetti Max - Lista attributi per oggetti Max specifici

3.1 SORGENTI PER LA SINTESI SOTTRATTIVA

Come sappiamo dal paragrafo 3.1 della teoria, lo scopo di un filtro è generalmente quello di modificare in qualche modo lo spettro di un segnale. Introduciamo quindi per prima cosa un oggetto MSP che ci serve per visualizzare lo spettro: **`spectroscope~`**. Questo oggetto grafico è reperibile nella sottocategoria "Audio" della *palette* "Add" (vedi fig. 3.1), ma se non lo trovate potete, come abbiamo già visto, prendere un *object box* e scrivere al suo interno il nome dell'oggetto.

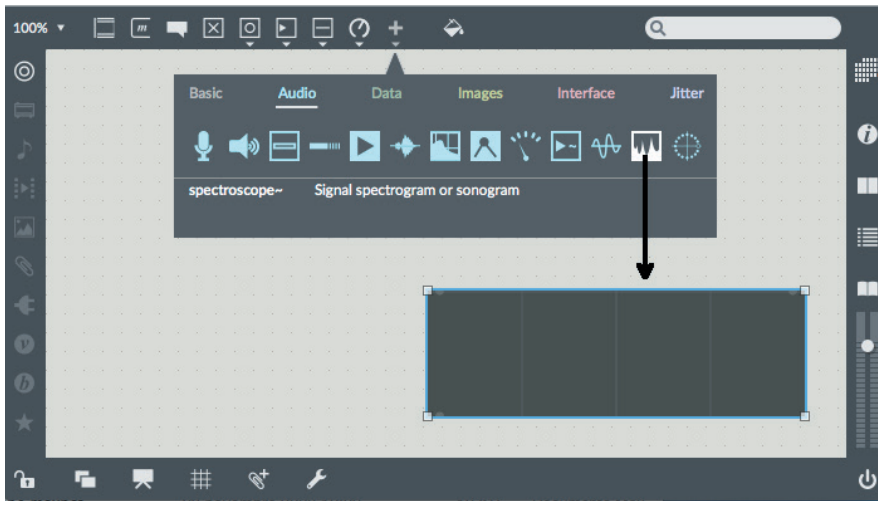


fig. 3.1: oggetto `spectroscope~`

Aprite ora il file **03_01_spectroscope.maxpat** (fig. 3.2).

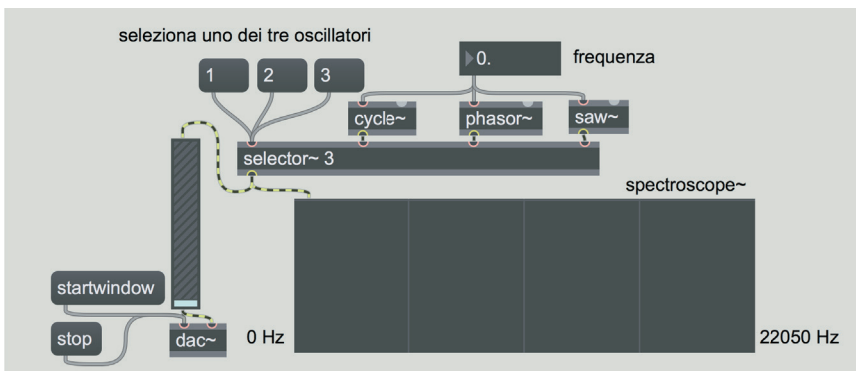


fig. 3.2: file 03_01_spectroscope.maxpat

Qui abbiamo collegato allo spettroscopio un oggetto `selector~` che ci permette di "smistare" tre oscillatori con forma d'onda sinusoidale (`cycle~`), a dente di sega non limitata in banda (`phasor~`) e a dente di sega limitata in

banda[1] (`saw~`). All'ingresso di sinistra di `selector~` (che abbiamo già visto al paragrafo 1.2) sono collegati tre *message box* che servono a selezionare uno dei tre oscillatori: impostando la frequenza nel *float number box* e selezionando i tre oscillatori possiamo vedere gli spettri relativi. Notate che la sinusoide contiene una sola componente, mentre all'oggetto `phasor~`, che genera una forma d'onda non limitata in banda, corrisponde lo spettro più ricco.[2]
Fate alcune prove, cambiando la frequenza e selezionando le diverse forme d'onda e osservate le immagini che si producono nello spettroscopio.
Come abbiamo detto quello che viene visualizzato è lo spettro del suono in ingresso: le diverse componenti del suono sono distribuite da sinistra a destra, e di default vengono visualizzate le frequenze da 0 Hz a 22050 Hz.
Questi due valori, ovvero la frequenza minima e massima che lo spettroscopio ci può mostrare, sono modificabili tramite l'*inspector* (categoria "Value", attributo "Lo and Hi Domain Display Value").
Provate ad aggiungere l'oggetto `spectroscope~` alle *patch* che avete già realizzato, in modo da familiarizzarvi con la relazione che c'è tra un suono e il suo contenuto spettrale: potete aggiungere l'oggetto anche alle *patch* dei capitoli precedenti, provate ad esempio con 01_14_audiofile.maxpat (collegando lo spettroscopio all'uscita di sinistra dell'oggetto `sfplay~`) oppure con IA_06_random_walk.maxpat (collegandolo all'uscita di [`p` monosynth]): in quest'ultima *patch* riuscite a cogliere la relazione tra la frequenza del suono e la forma dello spettro? (Provate il *preset* n. 5).
Passiamo adesso al rumore bianco, che viene generato in Max dall'oggetto `noise~` (vedi fig. 3.3).

fig. 3.3: generatore di rumore bianco

Nell'immagine (che vi invitiamo a ricostruire) abbiamo collegato il generatore di rumore all'oggetto `spectroscope~` tramite il quale possiamo vedere che lo spettro del rumore bianco contiene energia a tutte le frequenze. A differenza degli altri generatori di suono che abbiamo incontrato finora, il generatore di rumore bianco non ha bisogno di alcun parametro: la sua funzione infatti è

[1] Vedi paragrafo 1.2

[2] L'oggetto `spectroscope~` contiene un algoritmo di analisi spettrale denominato *Fast Fourier Transform* (Trasformata di Fourier Veloce): avevamo già accennato al teorema di Fourier alla fine del capitolo 2 della teoria e torneremo sull'argomento, con maggiori dettagli, nel capitolo 12.

generare un segnale costituito da valori casuali compresi tra -1 e 1 alla frequenza di campionamento (vedi par. 3.1 della teoria).
Il secondo tipo di generatore di rumore che abbiamo a disposizione con Max è l'oggetto `pink~` che genera rumore rosa (fig. 3.4)

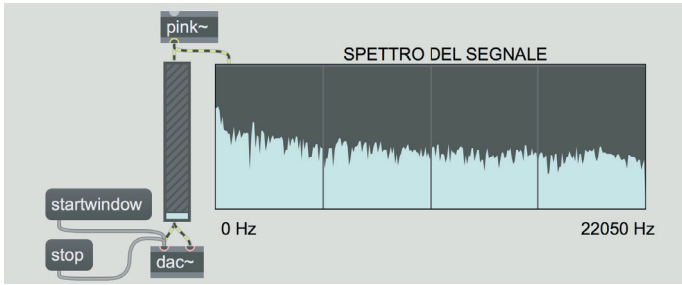


fig. 3.4: rumore rosa

Notate che, a differenza del rumore bianco, lo spettro del rumore rosa subisce un'attenuazione man mano che si procede verso le frequenze più alte, e l'attenuazione (come sappiamo dal par. 3.1 della teoria) è di 3 dB per ottava. Ricostruite la *patch* e ascoltate bene la differenza tra il rumore rosa e il rumore bianco: quale dei due suoni vi sembra più gradevole (o meno sgradevole), e perché?
Aggiungete alle due *patch* appena realizzate un oscilloscopio (`scope~`) di cui avrete impostato a 2 l'attributo "Calccount - samples per pixel"[3] nell'*inspector* (per trovare l'attributo andate alla categoria "Value"), e osservate le differenze tra la forma d'onda del rumore bianco e quella del rumore rosa.
In fig. 3.5 vediamo le due forme d'onda affiancate.

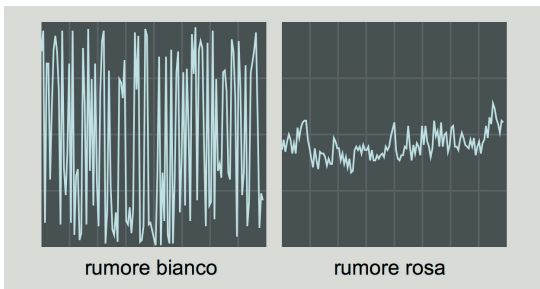


fig. 3.5: forma d'onda del rumore bianco e del rumore rosa

Senza entrare nei dettagli tecnici possiamo osservare che mentre il rumore bianco è, come sappiamo, una successione di valori casuali, il rumore rosa viene generato con un algoritmo più complesso, in cui un campione, che è sempre "casuale", non può però discostarsi eccessivamente dal precedente, e questo genera l'andamento "serpeggiante" della forma d'onda che vediamo in figura.
Il comportamento delle due forme d'onda corrisponde al loro contenuto spettrale: maggiore infatti è la differenza tra un campione e il successivo e

[3] Ne abbiamo parlato al paragrafo 1.2.

maggiore è l'energia alle frequenze più alte[4], e come abbiamo detto il rumore bianco ha maggiore energia alle frequenze alte rispetto al rumore rosa.

Un altro generatore interessante è **`rand~`** che genera campioni casuali ad una frequenza regolabile a piacere e li collega tramite linee, o meglio segmenti di retta (fig. 3.6). A differenza di `noise~` e `pink~`, che generano un nuovo campione casuale ad ogni ciclo del "motore" DSP (cioè generano ogni secondo un numero di campioni casuali pari alla frequenza di campionamento, ad es. 44100), con `rand~` è possibile stabilire la frequenza con cui i campioni casuali verranno generati, e il passaggio tra un campione e il successivo avviene gradualmente, tramite un'interpolazione lineare.

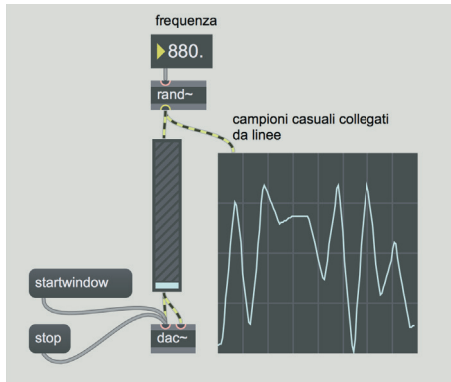


fig. 3.6: l'oggetto `rand~`

Questo generatore produce uno spettro che varia, come è ovvio, in relazione alla frequenza impostata, e che presenta una banda di frequenza principale che va da 0 Hz alla frequenza impostata, seguita da bande secondarie progressivamente attenuate la cui larghezza è pari alla frequenza impostata, vedi fig. 3.7.

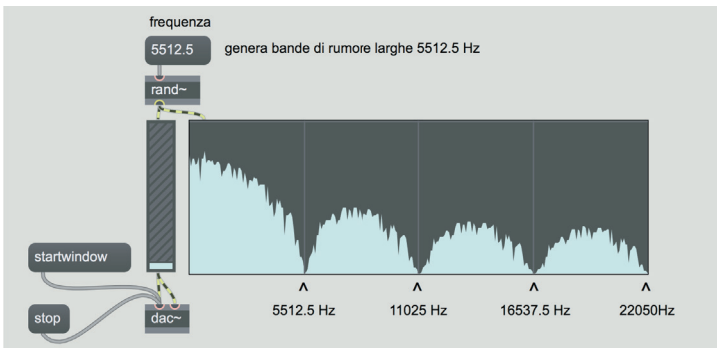


fig. 3.7: spettro generato dall'oggetto `rand~`

[4] Per comprendere questa affermazione osserviamo che la forma d'onda di un suono acuto oscilla velocemente, mentre quella di un suono grave oscilla lentamente. Nel primo caso, a parità di ampiezza, la differenza di valore tra due campioni successivi è mediamente maggiore che nel secondo caso.

Nell'esempio vediamo che la frequenza di **rand~** è di 5512.5 Hz (un quarto della massima frequenza visualizzabile nello spettroscopio in figura), e la prima banda va da 0 Hz a 5512.5 Hz. A questa seguono tre bande secondarie, progressivamente attenuate, tutte larghe 5512.5 Hz. Variando la frequenza di **rand~** si varia la larghezza delle bande e il loro numero: ad esempio se raddoppiamo la frequenza e la portiamo a 11025 Hz otteniamo due bande larghe appunto 11025 Hz.

Un altro generatore di rumore è **vs.rand0~**[5] (l'ultimo carattere prima della tilde è uno zero) che genera campioni casuali ad una frequenza data come **rand~**, ma non effettua alcuna interpolazione e mantiene il valore di ciascun campione fino alla generazione del campione successivo, creando un andamento a gradini.
Lo spettro è diviso in bande come lo spettro di **rand~** in fig. 3.7, ma l'attenuazione delle bande secondarie è molto minore a causa del brusco passaggio fra un campione e il successivo (vedi fig. 3.8).

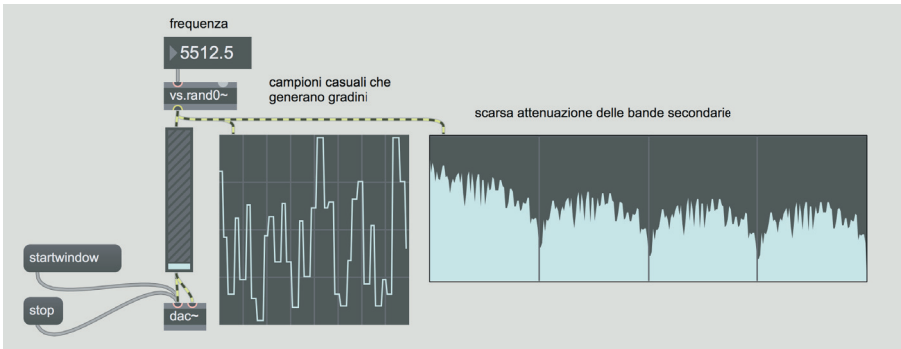


fig. 3.8: l'oggetto **vs.rand0~**

Nella libreria *Virtual Sound Macros* abbiamo anche un generatore di rumore con interpolazione cubica **vs.rand3~** (vedi fig. 3.9).

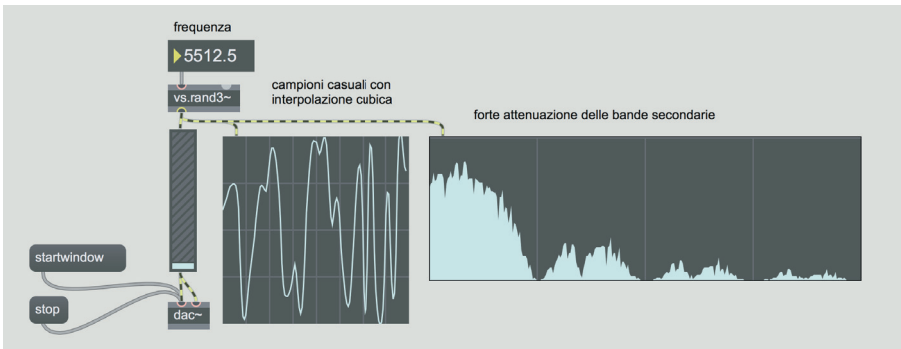


fig. 3.9: l'oggetto **vs.rand3~**

[5] Come si vede dal prefisso "vs" questo oggetto fa parte della libreria *Virtual Sound Macros*.

Come si vede nell'oscilloscopio, grazie all'interpolazione polinomiale il passaggio tra un campione e l'altro appare "smussato", essendo costituito non da una linea retta, ma da una curva: l'effetto è una forte attenuazione delle bande secondarie.
Ricreate le *patch* delle figure 3.6/3.9 per sperimentare i diversi generatori di rumore.
Altre sorgenti ricche di componenti con le quali si possono utilizzare efficacemente i filtri sono gli oscillatori "classici", con forma d'onda a dente di sega, quadrata e triangolare. Nel paragrafo 1.2 abbiamo visto i tre oscillatori limitati in banda che generano queste forme d'onda: `saw~`, `rect~` e `tri~`. Utilizzeremo questi oscillatori nel corso di questo capitolo.

Nel paragrafo 3.1 della parte teorica abbiamo anche accennato alla possibilità di filtrare i suoni campionati: daremo perciò, nel corso del capitolo, degli esempi di filtraggio di suoni campionati utilizzando l'oggetto `sfplay~` (di cui abbiamo parlato nel par. 1.5).

(...)

Il capitolo prosegue con:

3.2 FILTRI PASSA-BASSO, PASSA-ALTO, PASSA-BANDA ED ELIMINA-BANDA

3.3 IL FATTORE *Q* O FATTORE DI RISONANZA

3.4 GLI ORDINI DEI FILTRI E COLLEGAMENTO IN SERIE
L'oggetto umenu
Filtri del primo ordine
Filtri del secondo ordine
Filtri di ordine superiore: il collegamento in serie

3.5 LA SINTESI SOTTRATTIVA
Comunicazione "senza fili": l'oggetto pvar
Scelta multipla: l'oggetto radiogroup
Anatomia di un sintetizzatore in sintesi sottrattiva

3.6 L'EQUAZIONE DEI FILTRI DIGITALI
Il filtro non ricorsivo o filtro fir
Il filtro ricorsivo o filtro iir

3.7 FILTRI COLLEGATI IN PARALLELO ED EQUALIZZATORI GRAFICI
Utilizzare un banco di filtri paralleli
Equalizzatore grafico

3.8 ALTRE APPLICAZIONI DEL COLLEGAMENTO IN SERIE: EQUALIZZATORI PARAMETRICI E FILTRI SHELVING
Equalizzatore parametrico

3.9 ALTRE SORGENTI PER LA SINTESI SOTTRATTIVA: IMPULSI E CORPI RISONANTI

ATTIVITÀ
- Attività al computer: sostituzione di parti di algoritmi, correzione, completamento e analisi di algoritmi, costruzione di nuovi algoritmi

VERIFICHE
- Realizzazione di uno studio breve
- Compiti unitari di Reverse Engineering

SUSSIDI DIDATTICI
- Lista oggetti Max ·

Interludio B
ALTRI ELEMENTI DI PROGRAMMAZIONE CON MAX

CONTRATTO FORMATIVO

PREREQUISITI PER IL CAPITOLO

- Contenuti dei capp. 1, 2 e 3 (teoria e pratica), Interludio A

OBIETTIVI

Abilità

- Saper utilizzare diversi oggetti e segnali MIDI semplici
- Saper utilizzare operazioni ricorsive e conversioni in Max
- Saper costruire un arpeggiatore, anche con uso di intervalli probabilistici
- Saper smistare segnali e messaggi selezionando ingressi e uscite
- Saper confrontare valori e analizzarne la relazione
- Saper scomporre liste di dati
- Saper costruire sequenze ripetitive, tramite loop di dati
- Saper generare liste casuali per la simulazione di corpi risonanti
- Saper costruire uno Shepard Tone, o glissando infinito

CONTENUTI

- Uso di base del protocollo MIDI
- Operazioni ricorsive e sequenze ripetitive
- Arpeggiatori e intervalli probabilistici
- Confronto di valori, conversioni e smistamento di segnali e messaggi
- Scomposizione di liste e generazione di liste casuali
- Shepard Tone

TEMPI - Cap.3 (teoria e pratica) + Interludio B

Autodidatti

Per 300 ore globali di studio individuale (vol. I, teoria e pratica):

- ca. 110 ore

Corsi

Per un corso globale di 60 ore in classe + 120 di studio individuale (vol. I, teoria e pratica):

- ca. 18 ore frontali + 4 di feedback
- ca. 44 di studio individuale

ATTIVITÀ

Attività al computer:

- Attività al computer: sostituzione di parti di algoritmi, correzione, completamento e analisi di algoritmi, costruzione di nuovi algoritmi

VERIFICHE

- Compiti unitari di Reverse Engineering

SUSSIDI DIDATTICI

- Lista oggetti Max - Lista messaggi, attributi e parametri per oggetti Max specifici - Glossario

IB.1 CENNI SUL MIDI

Il MIDI è un sistema di comunicazione tra computer e strumenti musicali elettronici e/o digitali: tramite questo sistema è possibile, ad esempio, collegare (con un apposito cavetto MIDI) un computer ad un sintetizzatore e far sì che quest'ultimo venga "suonato" dal computer. In altre parole grazie al MIDI il computer può dire al sintetizzatore, tra le altre cose, quali note suonare, con che intensità, con che durata etc.
Gli strumenti musicali digitali, inoltre, possono anche essere "virtuali", cioè possono essere delle applicazioni residenti nel nostro computer che simulano il comportamento di strumenti digitali reali: anche con gli strumenti virtuali è possibile comunicare via MIDI, realizzando una connessione, in questo caso virtuale (cioè non con un cavo fisico), tra il programma che invia i comandi MIDI (ad esempio Max) e il programma (lo strumento virtuale) che li riceve. Come vedremo approfonditamente nel Cap. 9, Max ha diversi oggetti che sfruttano il protocollo MIDI. Dal momento che nel seguito di questo Interludio verranno utilizzati alcuni oggetti che gestiscono messaggi MIDI, forniamo qui una breve introduzione a questi oggetti, rimandando per i dettagli al capitolo 9 delle sezioni di teoria e pratica.
Aprite il file **IB_01_note_MIDI.maxpat**: in fig. IB.1 vediamo la parte superiore della *Patcher Window*.

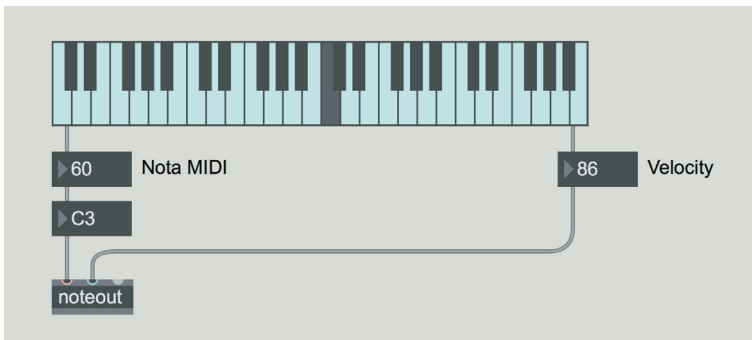


fig. IB.1: file IB_01_note_MIDI.maxpat, parte superiore

Abbiamo collegato l'oggetto **`kslider`** (la tastiera musicale) a dei *number box* e, tramite questi, all'oggetto **`noteout`**. Come possiamo vedere, facendo clic su un tasto di **`kslider`** avremo all'uscita di sinistra il valore MIDI della nota selezionata (cfr. anche il par. 1.4) e all'uscita di destra il valore di *velocity*, ovvero l'intensità della nota: facendo clic nella parte alta di un tasto di **`kslider`** si ottengono valori alti di *velocity*, facendo clic sulla parte bassa si ottengono valori bassi. La *velocity* può variare tra 1 e 127. I valori di nota MIDI e *velocity* vengono inviati agli ingressi di sinistra e centrale dell'oggetto **`noteout`**: quest'ultimo invia il comando relativo all'esecuzione della nota allo strumento (reale o virtuale) a cui è collegato. [1]

[1] L'ingresso di destra dell'oggetto **`noteout`** serve per impostare il canale MIDI, che al momento non ci serve: maggiori dettagli al cap. 9.

Nel protocollo MIDI questo comando si definisce "**note-on**". Se fate clic su un tasto del `kslider` (abbastanza in alto in modo da ottenere una *velocity* alta, superiore a 90) dovreste sentire un suono di pianoforte. Questo suono non proviene da Max, ma da uno strumento virtuale contenuto nel sistema operativo del vostro computer che per *default* è impostato sul suono del pianoforte. Se provate a suonare più note vi accorgerete che, per ogni coppia nota-*velocity* che l'oggetto `noteout` riceve, viene suonata una nuova nota, ma le precedenti non vengono interrotte: è come se i tasti fossero "incantati". Il problema è che, tramite `noteout`, stiamo dicendo allo strumento virtuale quando iniziare a suonare una nota, ma non gli diciamo quando interromperla!
Per interrompere una nota dobbiamo inviare nuovamente il valore MIDI relativo associato ad una *velocity* pari a 0. Il valore di *velocity* 0 corrisponde al comando "**note-off**", ovvero "solleva il dito dal tasto".
Per poter "spegnere" una nota MIDI tramite il `kslider` dobbiamo cambiare il modo con cui quest'ultimo gestisce i messaggi di nota MIDI: andate in modalità *edit* e richiamate l'*inspector* del `kslider` superiore, individuate la categoria "Value" e cambiate il menù a comparsa corrispondente al parametro "**Display Mode**" da "**Monophonic**" a "**Polyphonic**", infine tornate in modalità *performance*.
Adesso la prima volta che fate clic su un tasto del `kslider` verrà suonata una nota con la *velocity* corrispondente, un secondo clic sullo stesso tasto invierà nuovamente la nota, ma con *velocity* pari a 0, facendo terminare il suono: provate! Questo modo è definito "Polyphonic" perché, a differenza del "Monophonic", ci permette di tenere attive più note contemporaneamente.
In figura IB.2 vediamo la parte inferiore del file **IB_01_note_MIDI.maxpat**.

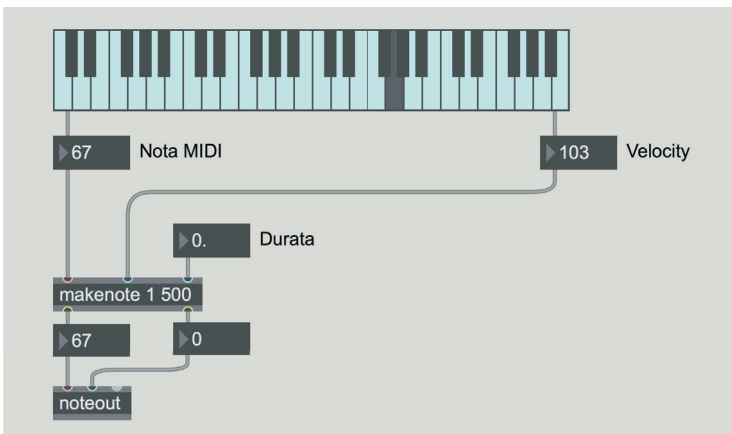


fig. IB.2: file IB_01_note_MIDI.maxpat, parte inferiore

Qui abbiamo collegato l'oggetto `kslider` (in modalità monofonica) all'oggetto **`makenote`**: quest'ultimo ogni volta che riceve un comando MIDI *note-on* genera il corrispondente comando MIDI *note-off* dopo un intervallo di tempo stabilito. L'oggetto ha tre ingressi, rispettivamente per il valore di nota MIDI, la *velocity* e la durata in millisecondi (ovvero per il tempo che deve passare tra un *note-on* e il successivo *note-off*), e due uscite, per il valore di nota MIDI e la *velocity*.

I parametri sono due, la *velocity* e la durata in millisecondi: nella *patch* abbiamo quindi una *velocity* pari a 1 e una durata di 500 millisecondi (mezzo secondo). Quando l'oggetto riceve un *note-on* lo invia direttamente alle uscite, dopo di che attende la durata prescritta (nel nostro caso 500 millisecondi) dopo di che invia il *note-off*. Notate che la *velocity* che inviamo tramite il `kslider` (nel caso in figura il valore 103) annulla e sostituisce il valore 1 che avevamo scritto come argomento: quest'ultimo ci è servito infatti solo per permetterci di scrivere il secondo argomento, cioè la durata, che in quanto "secondo argomento" deve essere per forza preceduto dal primo!
Anche la durata può essere modificata, inviando il nuovo valore all'ingresso di destra, che sostituirà il valore che abbiamo messo come secondo argomento.
Provate a suonare alcune note e a cambiare la durata nell'oggetto `makenote`: osservate come dalla sua seconda uscita venga generato prima un valore di *velocity* identico a quello generato dal `kslider` e, dopo il tempo stabilito, il valore 0.
Ora aggiungiamo un sommatore alla parte inferiore della *patch* nel modo illustrato in fig IB.3.

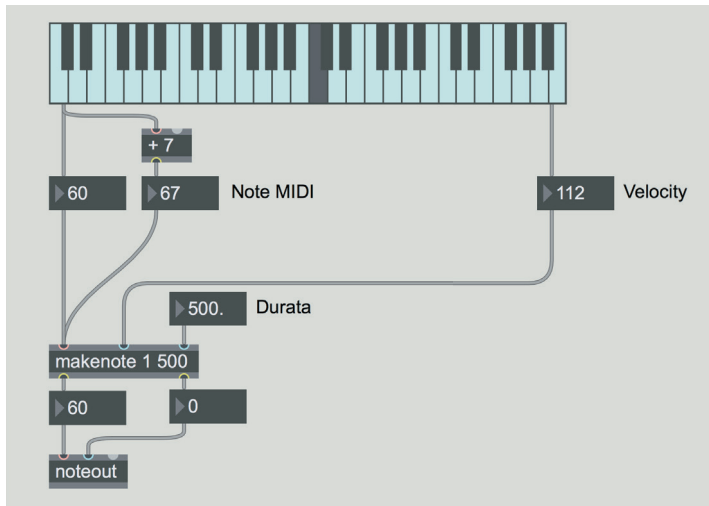


fig. IB.3: trasposizione MIDI

Questa *patch* è simile a quella del file **IA_01_trasposizione.maxpat** che abbiamo visto al primo paragrafo dell'interludio A. Anche in questo caso per ogni tasto premuto nell'oggetto `kslider` vengono generate due note a distanza di 7 semitoni, cioè di una quinta. Ogni volta che facciamo clic su un tasto di `kslider`, infatti, il valore di nota MIDI corrispondente viene inviato all'oggetto `makenote` e contemporaneamente ad un sommatore che aggiunge il valore 7 alla nota prima di inviarla a sua volta al `makenote`. Notate che per ottenere queste coppie di note non abbiamo avuto bisogno di sdoppiare anche il valore della *velocity*, né quello della durata: questi ultimi infatti corrispondono agli ingressi "freddi" dell'oggetto `makenote` e aggiornano soltanto le variabili interne dell'oggetto. Il contenuto di queste variabili interne viene riutilizzato ogni volta che un nuovo valore arriva all'ingresso "caldo": nel caso in figura, ad esempio, entrambe le note (DO e SOL centrali) hanno una *velocity* pari a 112 e una durata di 500 millisecondi.

Qui possiamo vedere che la regola secondo cui l'ingresso "caldo" di un oggetto è quello più a sinistra è il logico complemento della regola dell'ordine di esecuzione da destra a sinistra (cfr. par. 1.7): i primi messaggi ad essere trasmessi sono quelli più a destra, che raggiungono gli ingressi "freddi" e aggiornano le variabili interne di un oggetto (ad esempio la *velocity* in `makenote`), mentre gli ultimi sono quelli più a sinistra, che raggiungono un ingresso "caldo" e provocano un *output* dopo che tutte le variabili interne sono state aggiornate.

(...)

Il capitolo prosegue con:

IB.2 L'OPERATORE MODULO E LA RICORSIONE
La ricorsione
Costruiamo un arpeggiatore

IB.3 SMISTARE SEGNALI E MESSAGGI

IB.4 GLI OPERATORI RELAZIONALI E L'OGGETTO SELECT
L'oggetto select
Un metronomo "probabilistico"

IB.5 SCOMPORRE UNA LISTA, L'OGGETTO ITER

IB.6 LOOP DI DATI

IB.7 GENERARE UNA LISTA RANDOM

IB.8 CALCOLI E CONVERSIONI CON MAX
L'oggetto expr
Convertire intervalli di valori e segnali

IB.9 UTILIZZO DI TABELLE PER GLI INVILUPPI: LO SHEPARD TONE
Uso di tabelle per inviluppi consecutivi
Lo shepard tone

ATTIVITÀ

Attività al computer:

- Attività al computer: sostituzione di parti di algoritmi, correzione, completamento e analisi di algoritmi, costruzione di nuovi algoritmi

VERIFICHE

- Compiti unitari di Reverse Engineering

SUSSIDI DIDATTICI

- Lista oggetti Max - Lista J BPP>DDF; attributi e parametri per oggetti Max specifici - Glossario

4T
SEGNALI DI CONTROLLO

CONTRATTO FORMATIVO

PREREQUISITI PER IL CAPITOLO

- Contenuti dei capp. 1, 2, 3 (teoria)

OBIETTIVI

Conoscenze

- Conoscere la teoria e l'uso dei parametri degli oscillatori a bassa frequenza (LFO)
- Conoscere l'uso del DC Offset applicato agli LFO
- Conoscere l'uso della modulazione di frequenza per il vibrato
- Conoscere l'uso della modulazione di ampiezza per il tremolo
- Conoscere l'uso della modulazione del duty cycle (Pulse Width Modulation)
- Conoscere l'uso degli LFO per generare segnali di controllo per i filtri
- Conoscere l'uso di generatori di segnali pseudocasuali come LFO di controllo
- Conoscere l'uso degli oscillatori di controllo per lo spostamento del suono nei sistemi stereofonici e multicanale

Abilità

- Saper individuare all'ascolto le modifiche dei parametri base di un LFO e saperle descrivere

CONTENUTI

- Oscillatori a bassa frequenza: depth, rate e delay
- Gestione dei parametri degli LFO e uso del DC Offset
- Gestione del vibrato, del tremolo e del PWM mediante gli LFO
- Gestione dei parametri dei filtri mediante LFO
- Spostamento del suono nei sistemi stereo e multicanale
- Oscillatori di controllo modulati da altri LFO

TEMPI - Cap. 4 (Teoria e Pratica)

Autodidatti

Per 300 ore globali di studio individuale (vol. I, teoria e pratica):

- ca. 30 ore

Corsi

Per un corso globale di 60 ore in classe + 120 di studio individuale (vol. I, teoria e pratica):

- ca. 5 ore frontali + 1 di *feedback*
- ca. 12 di studio individuale

ATTIVITÀ

- Esempi interattivi

VERIFICHE

- Test a risposte brevi
- Test con ascolto e analisi

SUSSIDI DIDATTICI

- Concetti di base - Glossario

4.1 SEGNALI DI CONTROLLO: IL PANNING STEREOFONICO

Come abbiamo visto nel Cap. 1 è possibile variare i parametri di un suono (ad esempio la frequenza o l'ampiezza) tramite inviluppi che descrivono l'andamento nel tempo dei parametri in questione. I segnali (come quelli che controllano gli inviluppi), che non servono a generare un suono ma a variarne le caratteristiche, si definiscono **segnali di controllo**. Finora abbiamo usato, come segnali di controllo, soltanto segmenti di retta o di esponenziale. Questa tecnica si rivela efficace quando dobbiamo descrivere il cambiamento di un parametro attraverso pochi valori: ad esempio, per un inviluppo ADSR abbiamo bisogno di 4 segmenti, mentre il glissando di una nota può essere realizzato con una singola curva esponenziale. I parametri di un suono possono però variare anche in modo più complesso. Pensiamo ad esempio al vibrato di uno strumento ad arco: si tratta di un'oscillazione continua della frequenza della nota intorno ad un'altezza centrale. Per simulare questa vibrazione avremmo bisogno di decine o centinaia di segmenti, ma sarebbe evidentemente poco pratico e molto faticoso. Possiamo invece utilizzare un **oscillatore di controllo**, cioè un oscillatore che non serve a produrre suoni ma semplicemente valori che variano da un minimo a un massimo con una certa velocità e che vengono generalmente assegnati ai parametri degli *oscillatori audio*. Tali valori possono essere assegnati anche a parametri di altri algoritmi di sintesi ed elaborazione del suono.
È importante osservare che gli oscillatori di controllo sono *oscillatori in bassa frequenza* (**LFO, Low Frequency Oscillators**): essi cioè oscillano con frequenze generalmente inferiori a 30 Hz, e producono valori di controllo in continuo mutamento, che seguono la forma d'onda dell'oscillatore stesso. Ogni ampiezza istantanea dell'onda generata dall'oscillatore di controllo corrisponde ad un valore numerico che viene poi applicato ai parametri audio che vogliamo.
Facciamo un esempio: in figura 4.1 vediamo un LFO che controlla la posizione del suono nello spazio e genera una sinusoide che oscilla fra MIN (valore minimo) e MAX (valore massimo).

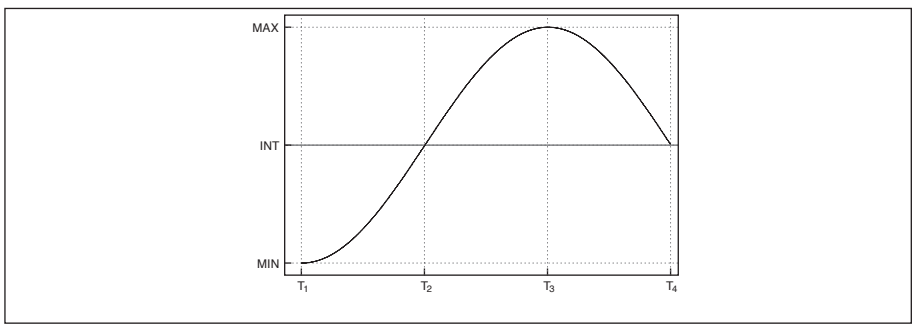


fig. 4.1: LFO che controlla la posizione del suono nello spazio

I valori minimo e massimo si riferiscono ai valori dell'*ampiezza* dell'oscillatore di controllo (di solito definita come **depth**, che in italiano significa profondità), la velocità dell'oscillazione dipende invece dalla *frequenza* dell'oscillatore di controllo (di solito definita come **rate**, che in italiano significa velocità, o frequenza).

I valori delle ampiezze istantanee della sinusoide generata da un LFO (o oscillatore di controllo) vengono usati come moltiplicatori delle ampiezze dei due canali in uscita di un oscillatore audio: quando la sinusoide dell'oscillatore di controllo raggiungerà il valore minimo MIN avremo il suono dell'oscillatore audio completamente a sinistra, quando l'oscillatore di controllo raggiungerà il valore massimo MAX avremo il suono dell'oscillatore audio completamente a destra, quando la sinusoide dell'oscillatore di controllo si troverà in corrispondenza del valore intermedio (INT) le ampiezze del suono sul canale sinistro e destro saranno uguali e di conseguenza percepiremo il suono al centro.
È ovviamente possibile anche utilizzare altre forme d'onda (triangolare, casuale etc.) per variare questi parametri di controllo; ad esempio, se usiamo un'onda quadra lo spostamento da destra a sinistra e viceversa non avverrà in modo continuo, come con la sinusoide, ma in modo alternato (MIN-MAX-MIN-MAX etc.).

ESEMPIO INTERATTIVO 4A • *Panning mediante LFO con diverse forme d'onda*

La velocità (cioè il *rate*) con cui tali valori oscillano dipende dalla frequenza che assegniamo all'oscillatore di controllo. Se utilizzassimo la frequenza 1 avremmo un'oscillazione fra MAX e MIN e ritorno una volta al secondo, se applicassimo una frequenza pari a .2 avremmo un'oscillazione completa ogni 5 secondi. E se utilizzassimo la frequenza 220? L'oscillazione sarebbe troppo veloce per poter percepire lo spostamento da destra a sinistra e ritorno (220 volte al secondo!); inoltre questa frequenza rientrerebbe nel campo audio e ciò aggiungerebbe nuove componenti allo spettro del suono risultante, come vedremo nel capitolo 10 nel paragrafo dedicato alla modulazione di ampiezza.

ESEMPIO INTERATTIVO 4B • *Panning mediante LFO sinusoidale a diverse frequenze*

Con gli oscillatori di controllo possiamo, in modi analoghi a quello descritto sopra, controllare ampiezza (*depth*) e velocità (*rate*) di un vibrato, di un tremolo, della variazione dei parametri di un filtro, come vedremo nei prossimi paragrafi.

(...)

Il capitolo prosegue con:

ATTIVITÀ
- Esempi interattivi

VERIFICHE
- Test a risposte brevi
- Test con ascolto e analisi

SUSSIDI DIDATTICI
- Concetti di base - Glossario

4P
SEGNALI DI CONTROLLO

CONTRATTO FORMATIVO

PREREQUISITI PER IL CAPITOLO

- Contenuti dei capp. 1, 2, 3 (teoria e pratica), cap.4 (teoria), Interludio A e B

OBIETTIVI

Abilità

- Saper far oscillare un suono nello spazio stereofonico
- Saper realizzare effetti di vibrato
- Saper simulare strumenti controllati in frequenza, come il theremin
- Saper realizzare effetti di tremolo
- Saper realizzare algoritmi di Pulse Width Modulation
- Saper variare in modo oscillante la frequenza di taglio, la frequenza centrale dei filtri, e il Q dei filtri mediante segnali di controllo
- Saper utilizzare segnali di controllo pseudocasuali
- Far ruotare (tramite un segnale di controllo) il suono in un sistema a 4 o più canali

Competenze

- Saper realizzare un breve studio sonoro basato sulle tecniche di controllo dei parametri mediante LFO

CONTENUTI

- Oscillatori a bassa frequenza: depth, rate e delay
- Gestione dei parametri degli LFO e uso del DC Offset
- Gestione del vibrato, del tremolo e del PWM mediante gli LFO
- Gestione dei parametri dei filtri mediante LFO
- Segnali di controllo pseudocasuali
- Spostamento del suono nei sistemi stereo e multicanale

TEMPI - Cap. 4 (Teoria e Pratica)

Autodidatti

Per 300 ore globali di studio individuale (vol. I, teoria e pratica):

- ca. 30 ore

Corsi

Per un corso globale di 60 ore in classe + 120 di studio individuale (vol. I, teoria e pratica):

- ca. 5 ore frontali + 1 di feedback
- ca. 12 di studio individuale

ATTIVITÀ

- Attività al computer: sostituzione di parti di algoritmi, correzione, completamento e analisi di algoritmi, costruzione di nuovi algoritmi

VERIFICHE

- Realizzazione di uno studio breve
- Compiti unitari di Reverse Engineering

SUSSIDI DIDATTICI

- Lista oggetti Max - Lista attributi per oggetti Max specifici - Glossario

4.1 SEGNALI DI CONTROLLO: IL PANNING STEREOFONICO

Per far oscillare un segnale nello spazio stereofonico, come descritto nel par. 4.1 della teoria, possiamo utilizzare, come segnale di controllo sinusoidale, il normale oggetto **`cycle~`** al quale daremo frequenze molto basse, al di sotto della minima frequenza udibile.
Abbiamo già visto come si definisce la posizione stereofonica di un segnale nel par. 1.6 [1]: facendo riferimento al file **01_18_pan_function.maxpat**, ricostruiamo in una nuova *patch* l'algoritmo che posiziona il suono nel fronte stereo (ovvero tutta la parte collegata all'oggetto **`line~`**, vedi fig. 4.1).

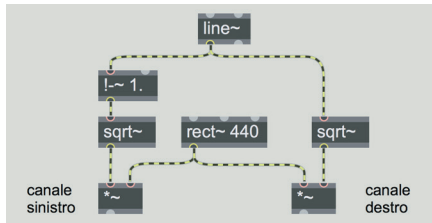


fig. 4.1: algoritmo per il *panning*

Dobbiamo ora sostituire **`line~`**, che modificava la posizione del suono tramite segmenti di retta, con il nostro segnale sinusoidale: l'oggetto **`cycle~`** però genera una sinusoide che oscilla tra -1 e 1, mentre a noi serve un segnale che oscilli tra 0 e 1 (quando il segnale è 0 il suono è posizionato a sinistra e quando è 1 è posizionato a destra). Si potrebbe modificare l'intervallo di oscillazione di **`cycle~`** con un paio di semplici calcoli, ma questo lo vedremo nel prossimo paragrafo; qui preferiamo usare un oggetto di cui abbiamo già parlato nell'Interludio B, al paragrafo IB.8. Completate la *patch* come da figura 4.2.

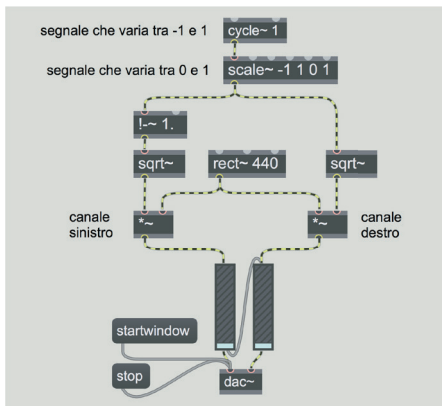


fig. 4.2: *panning* stereofonico controllato da un LFO

[1] Se non vi ricordate come si fa, rinfrescatevi la memoria rileggendo i paragrafi relativi nella parte di teoria e in quella di pratica.

Abbiamo sostituito all'oggetto **line~** l'oggetto **scale~** a cui abbiamo collegato un **cycle~**. Come sappiamo l'oggetto **scale~** ha 4 argomenti, i primi due specificano l'intervallo in entrata e gli ultimi due l'intervallo in uscita: nel nostro caso, gli argomenti [-1 1 0 1] indicano che se mandiamo a **scale~** un segnale che varia tra -1 e 1, avremo in uscita un segnale riscalato che varia tra 0 e 1, che è esattamente quello che ci serve. L'oggetto **cycle~** genera una sinusoide di controllo alla frequenza di 1 Hz, il suono fa quindi un "viaggio" dal canale sinistro al destro e ritorno nel tempo di un secondo: collegando un *float number box* a **cycle~** possiamo variare la frequenza di oscillazione.
Provate con frequenze diverse, ma non superiori ai 20 Hz: le frequenze superiori generano fenomeni di modulazione che tratteremo nel cap. 10.
Possiamo semplificare la *patch* usando l'oggetto **vs.pan~**, della libreria *Virtual Sound Macros*, che realizza l'algoritmo di *panning* prendendo un suono dall'ingresso sinistro e spostandolo nel fronte stereo secondo il segnale di controllo ricevuto all'ingresso destro (fig. 4.3).

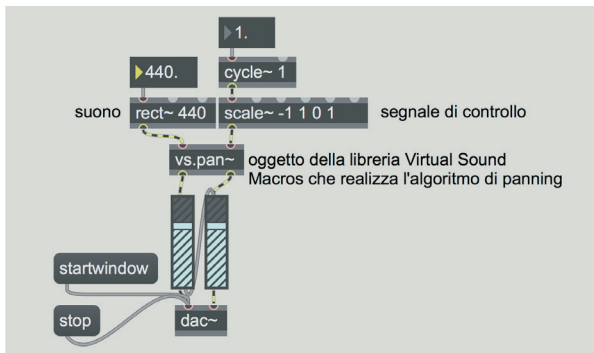


fig. 4.3: *panning* stereofonico con l'oggetto **vs.pan~**

Come si vede l'oggetto **vs.pan~** funziona come l'algoritmo di fig. 4.1, ma permette di "liberare spazio" nella nostra *patch*.

Possiamo usare come segnale di controllo un'altra forma d'onda, ad esempio la quadrata (fig. 4.4).

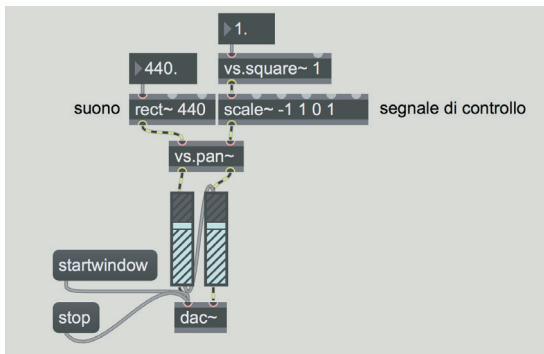


fig. 4.4: controllo del *panning* con LFO a forma d'onda quadra

In questo caso il suono si sposta da un canale all'altro senza passare per posizioni intermedie: questa discontinuità genera un clic indesiderato che può essere eliminato filtrando il segnale di controllo con un filtro passa-basso che serve a "smussare" gli spigoli dell'onda quadra (fig. 4.5).

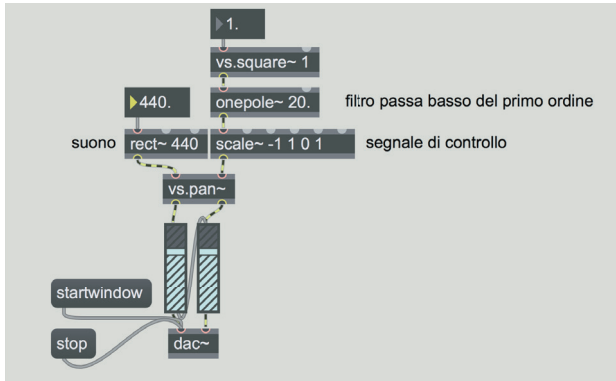


fig. 4.5: filtraggio di un LFO

Qui abbiamo impostato una frequenza di taglio di 20 Hz: questo significa, grosso modo, che il segnale di controllo non può "saltare" da un valore all'altro in un tempo inferiore a 1/20 di secondo. Provate a variare la frequenza di taglio del filtro per sentirne l'influenza sul percorso del suono: più la frequenza di taglio è bassa e più graduale è il passaggio da un canale all'altro.

(...)

Il capitolo prosegue con:

4.3 SEGNALI DI CONTROLLO PER LA FREQUENZA
Simuliamo un theremin

4.4 SEGNALI DI CONTROLLO PER L'AMPIEZZA

4.5 MODULAZIONE DEL DUTY CYCLE (PULSE WIDTH MODULATION)

4.6 SEGNALI DI CONTROLLO PER I FILTRI

4.7 ALTRI GENERATORI DI SEGNALI DI CONTROLLO
La matrice di modulazione

4.8 SEGNALI DI CONTROLLO: IL PANNING MULTICANALE

ATTIVITÀ

- Attività al computer: sostituzione di parti di algoritmi, correzione, completamento e analisi di algoritmi, costruzione di nuovi algoritmi

VERIFICHE

- Realizzazione di uno studio breve
- Compiti unitari di Reverse Engineering

SUSSIDI DIDATTICI

- Lista oggetti Max - Lista attributi per oggetti Max specifici - Glossario

Alessandro Cipriani • Maurizio Giri

Musica Elettronica e Sound Design

Teoria e Pratica con Max 7 • volume 1

Argomenti trattati

Sintesi ed Elaborazione del Suono - Frequenza, Ampiezza e Forma d'Onda - Inviluppi e Glissandi - Sintesi Additiva e Sintesi Vettoriale - Sorgenti di Rumore - Filtri - Sintesi Sottrattiva - Realizzazione di Sintetizzatori Virtuali - Equalizzatori, Impulsi e Corpi Risonanti - Segnali di Controllo e LFO - Tecniche di Programmazione con Max e MSP

"Il libro di Alessandro Cipriani e Maurizio Giri costituisce uno dei primi corsi di musica elettronica che integra esplicitamente percezione, teoria e pratica, usando esempi di sintesi in tempo reale che si possono manipolare e personalizzare. Secondo il mio parere, Cipriani e Giri hanno compiuto un lavoro magistrale consentendo all'esperienza e alla conoscenza teorica di rinforzarsi l'una con l'altra. Questo libro funziona sia come testo all'interno di un corso, sia come mezzo per l'autoapprendimento. In aggiunta, il libro include un'introduzione completa all'elaborazione digitale dei segnali con Max e costituisce una splendida introduzione ai concetti di programmazione con questo software. Spero che trarrete vantaggio dagli eccellenti esempi di Max creati dagli autori: sono insieme divertenti e illuminanti, e suonano abbastanza bene da poter essere utilizzati anche sul palco. Vale la pena esaminarli come modelli o per estenderli in modi sempre nuovi." (dalla prefazione di **David Zicarelli**)

Questo è il primo volume di un sistema didattico organico in tre volumi. Ad ogni capitolo di teoria corrisponde un capitolo di pratica con il software Max (uno dei più potenti e affidabili software per l'elaborazione del suono in tempo reale, per Windows e Mac OSX) e una sezione online: in questo modo lo studente acquisisce conoscenze, abilità e competenze teorico-pratiche in modo integrato. Il testo può essere studiato autonomamente oppure sotto la guida di un insegnante. È ideale quindi per chi inizia da zero, ma utilissimo anche per chi voglia approfondire la propria competenza nel campo del sound design e della musica elettronica.

ALESSANDRO CIPRIANI è coautore del testo "Virtual Sound" sulla programmazione in Csound. Le sue composizioni sono state eseguite nei maggiori festival internazionali di musica elettronica e pubblicate da Computer Music Journal, International Computer Music Conference etc. Ha scritto musiche per il Teatro dell'Opera di Pechino e per film e documentari in cui ambienti sonori, dialoghi e musica, elaborati al computer, si fondono ed hanno funzioni interscambiabili. Ha tenuto seminari in numerose università europee e americane (University of California, Sibelius Academy Helsinki, Conservatorio Tchaikovsky di Mosca, etc.). È titolare della Cattedra di Musica Elettronica del Conservatorio di Frosinone e socio fondatore di Edison Studio. È membro dell'Editorial Board della rivista Organised Sound (Cambridge University Press).

MAURIZIO GIRI è docente in Composizione ed insegna tecniche di programmazione con Max nei Conservatori di Perugia e Frosinone. Ha scritto musica strumentale ed elettroacustica. Attualmente si occupa di musica elettronica e nuove tecnologie applicate all'elaborazione digitale del suono, all'improvvisazione e alla composizione musicale. Ha scritto software di composizione algoritmica, improvvisazione elettroacustica e live electronics. Ha fondato la società Amazing Noises, che sviluppa applicazioni musicali e plug-in per dispositivi mobili e computer tradizionali. Ha pubblicato tutorial su Max in riviste specializzate. È stato artista residente a Parigi (Cité Internationale des Arts) e a Lione (GRAME). Ha collaborato con l'Institut Nicod alla École Normale Supérieure di Parigi ad un progetto di filosofia del suono.

ISBN 978-88-99212-00-1
9 788899 212001

www.contemponet.com

€ 45.00 iva inclusa